A New Order-theoretic Characterisation of the Polytime Computable Functions[*]

Martin Avanzini[1], Naohi Eguchi[2] and Georg Moser[1]

[1] Institute of Computer Science, University of Innsbruck, Austria

{martin.avanzini,georg.moser}@uibk.ac.at

[2] Mathematical Institute Tohoku University, Japan

{eguchi@math.tohoku.ac.jp

January 18, 2012

We propose a new order, the *small polynomial path order* (*sPOP** for short). The order sPOP* provides a characterisation of the class of polynomial time computable function via term rewrite systems. Any polynomial time computable function gives rise to a rewrite system that is compatible with sPOP*. On the other hand any function defined by a rewrite system compatible with sPOP* is polynomial time computable.

Technically sPOP* is a tamed recursive path order with product status. Its distinctive feature is the precise control provided. For any rewrite system that is compatible with sPOP* that makes use of recursion up to depth d, the (innermost) runtime complexity is bounded from above by a polynomial of degree d.

1 Introduction

In this paper we are concerned with the *complexity analysis* of term rewrite systems (TRSs) and the ramifications of such an analysis in *implicit computational complexity.*

Term rewriting is a conceptually simple, but powerful abstract model of computation. The foundation of rewriting is equational logic and *term rewrite systems* (*TRSs* for short) are conceivable as sets of directed equations. The implicit orientation of equations in TRSs naturally gives rise to computations, where a term is rewritten by successively replacing subterms by equal terms until no further reduction is possible. Such a sequence of rewrite steps is also called a *derivation.*

A natural way to measure the complexity of a TRS $\mathcal{R}$ is to measure the length of computations in $\mathcal{R}$. More precisely the *runtime complexity* of a TRS relates the maximal lengths

[*] This work is partially supported by FWF (Austrian Science Fund) project I-608-N18 and by a grant of the University of Innsbruck.

of derivations to the size of the initial term. Furthermore the shape of the initial term is suitable restricted. The latter restrictions aims at capturing the complexity of the *functions computed* by the analysed TRS. Indeed, the runtime complexity of a TRS $\mathcal{R}$ forms an *invariant cost model*. Suppose the runtime complexity of $\mathcal{R}$ is polynomially bounded and the function computed by $\mathcal{R}$ is implemented on a Turing machine. Then the runtime of this Turing machine is polynomially bounded [3].

We propose a new order, the *small polynomial path order* (*sPOP** for short). The order sPOP* provides a characterisation of the class of *polynomial time computable function* (*polytime computable functions* for short) via term rewrite systems. Any polytime computable function gives rise to a rewrite system that is compatible with sPOP*. On the other hand any function defined by a rewrite system compatible with sPOP* is polytime computable. The proposed order embodies the principle of *predicative recursion* as proposed by Bellantoni and Cook [7]. Our result bridges the subject of (automated) complexity analysis of rewrite systems and the field of implicit computational complexity (*ICC* for short).

Our results entail a new syntactic criteria to automatically establish polynomial runtime complexity of a given TRS. This criteria extends the state of the art in runtime complexity analysis as it is more precise or more efficient than related techniques. Note that the analysis is automatic as for any given TRS, compatibility with sPOP* can be efficiently checked by a machine. Should this check succeeds we get an asymptotic bound on the runtime complexity directly from the parameters of the order. It should perhaps be emphasised that compatibility of a TRS with sPOP* implies termination and thus our complexity analysis technique do not presuppose termination, instead we use (variations) of termination techniques to induce upper bounds on the complexity.

Our syntactic account of predicative recursion delineates a class of rewrite systems: a rewrite system $\mathcal{R}$ is called *predicative recursive of degree d* if $\mathcal{R}$ is compatible with sPOP* and the depth of recursion of all function symbols in $\mathcal{R}$ is bounded by d (see Section 4 for the formal definition). Any predicative recursive rewrite system of degree d admits runtime complexity in $O(n^d)$.

1.1 Related works

Polynomial runtime complexity analysis is an active research area in rewriting. Interest in this field greatly increased recently. This is partly due to the incorporation of a dedicated category for complexity into the annual termination competition (TERMCOMP).[1] We mention very recent work on matrix interpretations that is readily applicable to runtime complexity analysis by Middeldorp et al. [21] and recent work on the incorporation of the dependency pair method in complexity analysis [13, 14, 23]. See [22] for an overview on work on complexity analysis in rewriting. The most powerful techniques for runtime complexity analysis currently available, basically employ semantic considerations on the rewrite systems, which are notoriously inefficient.

There are several accounts of predicative analysis of recursion in the (ICC) literature. We mention only those related works which are directly comparable to our work. See [5] for an overview on ICC. Notable the clearest connection of our work is to Marion's *light multiset path order* (*LMPO* for short) [18]. This path order forms a strict extension of the here

[1] `http://termcomp.uibk.ac.at/`.

proposed order sPOP*, but lacks the precision of the latter. In Bonfante et. al. [8] restricted classes of polynomial interpretations are studied that can be employed to obtain similar precise polynomial upper bounds on the runtime complexity of TRSs as with sPOP*. Neither of these results is applicable to relate the depth of recursion to the runtime complexity, in the sense mentioned above. We have also drawn motivation from [20] which provides a related fine-grained capturing of the polytime computable functions, but which lacks applicability in the context of runtime complexity analysis.

Above we emphasised that our analysis is automatic and there are other recent approaches for the automated analysis of resource usage in programs. Notable Hoffmann et al. [16] provide an automatic multivariate amortised resource analysis which extends earlier results on an automatic cost analysis using typing. Albert et al. [1] present an automated complexity tool for Java Bytecode programs and Gulwani et al. [11] provide an automated complexity tool for C programs.

1.2 Contributions

1.2.1 Precise Runtime Complexity Analysis

The proposed order sPOP* is a restriction of the *polynomial path order* (*POP** for short) introduced by the second and third author [2]. Crucially sPOP* is a tamed recursive path order with product status [4]. Its distinctive feature is the precise control provided for runtime complexity analysis: for any predicative recursive TRS of degree d its runtime complexity lies in $O(n^d)$ (cf. Theorem 2). Furthermore this bound is tight, that is, we provide a family of TRSs, delineated by sPOP*, whose runtime complexity is bounded from below by $\Omega(n^d)$.

1.2.2 Fine-Grained Capture of Polytime Functions

As already mentioned the runtime complexity of a TRS forms an invariant cost model. Hence sPOP* is *sound* for the class of polytime computable functions: any function f computable by a TRS $\mathcal{R}$, such that $\mathcal{R}$ is compatible with sPOP* is polytime computable. On the other hand sPOP* is *complete*: for any polytime computable function f there exists a TRS $\mathcal{R}$ computing f such that $\mathcal{R}$ is compatible with sPOP*. However, for runtime complexity, we can obtained a more fine-grained classification. We establish that those TRS that are definable with d nestings of predicative recursion are predicative recursive of degree d (cf. Theorem 13). Conclusively the runtime complexity of these systems lies in $O(n^d)$. Thereby we obtain a fine-grained characterisation of the polytime computable functions, which may be of interest in implicit computational complexity theory.

1.2.3 Parameter Substitution

We extend upon sPOP* by proposing a generalisation of sPOP*, admitting the same properties as above, that allows to handle more general recursion schemes that make use of parameter substitution (cf. Theorem 22). As a corollary to this and the fact that the runtime complexity of a TRS forms an invariant cost model we conclude a non-trivial closure property of the class $\mathcal{B}_{\mathsf{wsc}}$: this class is closed under predicative recursion with parameter substitution.

1.2.4 Automated Complexity Analysis

We have implemented the order sPOP* in the *Tyrolean Complexity Tool* T$_\mathsf{C}$T, version 1.9, an open source complexity analyser.[2] The experimental evidence obtained indicates the viability of the method.

2 Motivation

We present the main ideas of the proposed *small polynomial path order* and provide an informal account of the technical results obtained in the remainder of the paper.

The order sPOP* essentially embodies the predicative analysis of recursion set forth by Bellantoni and Cook. In [7] a recursion-theoretic characterisation $\mathcal{B}$ of the class of polytime computable functions is proposed. This analysis is connected to the important principle of *tiering* introduced by Simmons [24] and Leivant [17]. The essential idea is that the arguments of a function are separated into *normal* and *safe* arguments (or correspondingly into arguments of different tiers).

It is a natural idea to seek out term-rewriting characterisations of the polytime computable functions [6]. Indeed, Beckmann and Weiermann successfully apply their characterisation to yield a non-trivial closure property of the class $\mathcal{B}$. Given a TRS $\mathcal{R}$ representing a polytime computable function, we seek syntactic criteria on $\mathcal{R}$ to verify that the runtime complexity of $\mathcal{R}$ lies in $O(n^d)$ for $d \in \mathbb{N}$.

Let us make this idea precise. We present a subclass $\mathcal{B}_{\mathsf{wsc}}$ of $\mathcal{B}$ that only induces TRSs compatible with sPOP*. We formulate the class $\mathcal{B}_{\mathsf{wsc}}$ over the set $\{0,1\}^*$ of binary words, where we write ϵ to denote the empty sequence and $S_i(;x)$ to denote the word xi. We are assuming that the arguments of every function are partitioned in to normal and safe ones. Notationally we write $f(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$ where argument are separated by a semicolon. Normal arguments are always drawn to the left, and safe arguments to the right of the semicolon. The class $\mathcal{B}_{\mathsf{wsc}}$ is the smallest class containing certain initial functions and closed under *weak safe composition* and *safe recursion on notation*, which are presented in Fig. 1. Due to a variation of a result by Handley and Wainer, we have that $\mathcal{B}_{\mathsf{wsc}}$ captures the polytime functions [12]. A term-rewriting characterisation of the polytime functions is obtained by orienting the equations in Fig. 1 from left to right.

Suppose the definition of $\mathcal{R}$ is based on the equations in $\mathcal{B}_{\mathsf{wsc}}$. It seems likely to deduce a precise bound on the runtime complexity of $\mathcal{R}$ by measuring the number of nested applications of safe recursion. We establish that such a TRS is predicative recursive of degree d, where d is the maximal nesting of schema (SRN) (cf. Theorem 13). This result is based on a suitable definition of sPOP*: the parameters and order constraints present reflect the operators in the class $\mathcal{B}_{\mathsf{wsc}}$.

In order to employ the separation of normal and safe arguments, we fix for each defined symbol a partitioning of argument positions into *normal* and *safe* positions. For constructors we fix that all argument positions are safe. Moreover sPOP* restricts recursion to normal argument. Dually only safe argument positions allow the substitution of recursive calls. Via the order constraints we can also guarantee that only normal arguments are substituted for normal argument positions. Hence sPOP* enforces a weak composition schema for function

[2] Available at `http://cl-informatik.uibk.ac.at/software/tct`.

$$
\begin{array}{lll}
\textbf{Initial Functions} & P(;\epsilon) = \epsilon & \\
 & P(;S_i(;x)) = x & (i = 0,1) \\
 & I^{k,l}_j(\vec{x};\vec{y}) = x_j & (j \in \{1,\dots,k\}) \\
 & I^{k,l}_j(\vec{x};\vec{y}) = y_{j-k} & (j \in \{k+1,\dots,l+k\}) \\
 & C(;\epsilon,y,z_0,z_1) = y & \\
 & C(;S_i(;x),y,z_0,z_1) = z_i & (i = 0,1) \\
 & O(\vec{x};\vec{y}) = \epsilon & \\
\textbf{Weak Composition } (\mathsf{WSC}) & f(\vec{x};\vec{y}) = h(x_{i_1},\dots,x_{i_k};\vec{g}(\vec{x};\vec{y})) & \\
\textbf{Safe Recursion } (\mathsf{SRN}) & f(\epsilon,\vec{x};\vec{y}) = g(\vec{x};\vec{y}) & \\
 & f(S_i(;z),\vec{x};\vec{y}) = h_i(z,\vec{x};\vec{y},f(z,\vec{x};\vec{y})) & (i = 0,1)
\end{array}
$$

Figure 1: Defining initial functions and operations for $\mathcal{B}_{\mathsf{wsc}}$

$$\begin{tikzcd}
s_1 \arrow[d] & \xrightarrow{\mathsf{i}}_{\mathcal{R}} & s_2 \arrow[d] & \xrightarrow{\mathsf{i}}_{\mathcal{R}} & \cdots & \xrightarrow{\mathsf{i}}_{\mathcal{R}} & s_\ell \arrow[d] \\
\mathsf{S}(s_1) & \blacktriangleright_k & \mathsf{S}(s_2) & \blacktriangleright_k & \cdots & \blacktriangleright_k & \mathsf{S}(s_\ell)
\end{tikzcd}$$

Figure 2: Embedding of the innermost rewrite relation into $\blacktriangleright_k$.

composition, as well as a safe recursion schema for recursion. For the latter the comparison of arguments via a product status, rather than via a multiset status is also essential. The formal definition of sPOP* is given in Section 4.

We remark on the connection between sPOP* and other path orders. It is clear that an order-theoretic characterisation of predicative recursion is obtained as a restriction of the recursive path order. Predicative recursion stems from a careful analysis of primitive recursion and the recursive path order (with multiset status) characterises the class of primitive recursive functions [15]. The light multiset path order proposed by Marion is based on the separation of normal and safe arguments and adapts function composition suitable to retain the separation of normal and arguments. However, the composition schema goes beyond weak composition. It is shown in [18] that LMPO captures the polytime computable functions, but this result relies on a clever use of memoisation techniques. In particular the aforementioned restrictions are not enough to forbid the treatment of TRSs that induce non-feasible runtime complexity. To recover the situation, POP* additionally requires the absence of multiple recursive calls in a rule. Then it can be shown that in fact compatible TRSs admits feasible runtime complexity [2]. However, for rewrite systems of predicative recursion of degree d, POP* overestimates the runtime complexity.

To show that sPOP* is correct, we make use of a variety of ingredients. First in Section 4 we propose a family of TRSs that establish the tightness of the obtained bound. Then in Section 5, we define *predicative interpretations* S that flatten terms to *sequences of terms*, essentially separating safe from normal arguments. This allows us to analyse a term independent from its safe arguments. In Section 6 we introduce an order $\blacktriangleright_k$ on sequences of terms, that is simpler compared to $>_{\mathsf{spop}*}$ and does not rely on the separation of argument positions. In Section 7 we show that predicative interpretations S embeds innermost rewrite steps into $\blacktriangleright_k$ as depicted in Fig. 2. As the length of $\blacktriangleright_k$ descending sequences starting from basic terms can be bound appropriately (cf. Theorem 9), we obtain correctness. Finally, to show that sPOP* is complete, it suffices to show that any $\mathcal{B}_{\mathsf{wsc}}$ only induces TRSs compatible with sPOP*(cf. Section 8).

3 Preliminaries

We assume at least nodding acquaintance with the basics of rewriting cf. [4]. In this section we fix the bare essential of notions and notation, we use in the remainder of the paper.

Throughout the paper, we fix a countably infinite set of *variables* $\mathcal{V}$ and a finite *signature* $\mathcal{F}$ consisting of *function symbols* $f, g, h, \dots$. The signature associates with each function symbol f a natural number, its *arity*. The set of *terms* is denoted as $\mathcal{T}(\mathcal{F},\mathcal{V})$ (or $\mathcal{T}$ for short). We write $s \trianglerighteq t$ to indicate that s is a *subterm* of t and $s \vartriangleright t$, if s is a *proper subterm*. The *size* of t refers to the number of function symbols and variables it contains. The *depth* $\mathsf{dp}(t)$ of t is 0 if t is a variable or a constant, for $t = f(t_1,\dots,t_n)$ the depth of t is $d+1$ where d is the maximal depth of an argument $t_1,\dots,t_n$.

A precedence $\succcurlyeq$ is a preorder on the set of function symbols $\mathcal{F}$. As usual $\succcurlyeq$ induces an equivalence $\sim$ and a strict proper order $\succ$. If $f \succ g$ we say that g is *(strictly) below* f in the precedence. This will indicate that the definition of f depends on g. In this sense $\succcurlyeq$ can be seen as a form of static call graph. The *rank* $\mathsf{rk}(f) = 1 + \max\{\mathsf{rk}(g) \mid f \succ g\}$ measures the length of $\succ$ chains starting from the function symbol f. (We employ the convention that the maximum of an empty set equals 0.) The equivalence $\sim$ is lifted from function symbols to terms by additionally disregarding the order on arguments. Formally s and t are *equivalent*, in notation $s \sim t$, if $s = t$, or $s = f(s_1,\dots,s_n)$ and $t = g(t_1,\dots,t_n)$ where $f \sim g$ and $s_i \sim t_{\pi(i)}$ for all arguments and some permutation π.

A *rewrite rule* is a pair (l,r) of terms, denoted as $l \to r$, where the *left-hand side* l is not a variable and the *right-hand side* r mentions only variables from l. A *term rewrite system* (*TRS* for short) $\mathcal{R}$ (over the signature $\mathcal{F}$) is a *finite* set of rewrite. The root symbols of left-hand sides are called *defined symbols*, the remaining function symbols are called *constructors*. A term that only contains constructors is also called *value*, and the set of all values is denoted by $\mathcal{V}\mathsf{al}$.

Let $\mathcal{R}$ denote a TRS. The TRS $\mathcal{R}$ induces the *rewrite relation* $\to_{\mathcal{R}}$ on terms as follows. Informally, a term s *rewrites* to t if the left-hand side l of a rewrite rule $l \to r$ from $\mathcal{R}$ matches some subterm of s, and the term t is obtained from s by replacing the matched subterm with the corresponding instance of the right-hand side r. Formally, $s \to_{\mathcal{R}} t$ if there exists a *context* C, *substitution* σ and rule $l \to r \in \mathcal{R}$ such that $s = C[l\sigma]$ and $t = C[r\sigma]$. Here a context C is a term with exactly one occurrence of a *hole* $\Box$, and $C[t]$ denotes the term obtained by replacing the hole $\Box$ in C by t. A substitution σ is a function that maps variables to terms, and $t\sigma$ denotes the homomorphic extension of this function to terms.

A term t is called a *normal form* (with respect to $\mathcal{R}$) if it is irreducible, ie., if there exists no term s with $t \to_{\mathcal{R}} s$. Consider the rewrite step $s \to_{\mathcal{R}} t$ where $s = C[l\sigma]$ as above. If all arguments of $l\sigma$ are normal forms, then this step is an *innermost rewrite step* and denoted by $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$. The relation $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ is called the *innermost rewrite relation* of $\mathcal{R}$. It requires arguments to be evaluated first, and can be seen as the adoption of a call-by-value semantics. In the sequel we will only be concerned with innermost rewriting.

The TRS $\mathcal{R}$ is a *constructor TRS* if arguments of left-hand sides only contain constructors. It is *completely defined* if values coincide with normal forms, that is, defined symbols do not occur in normal forms. The TRS $\mathcal{R}$ is called *terminating* if $\rightarrow_{\mathcal{R}}$ is well-founded, and $\mathcal{R}$ is *confluent* if all peaks $t_1 \ {}_{\mathcal{R}}^{*}\!\!\leftarrow s \rightarrow_{\mathcal{R}}^{*} t_2$ can be joined: $t_1 \rightarrow_{\mathcal{R}}^{*} u \ {}_{\mathcal{R}}^{*}\!\!\leftarrow t_2$ for some term u. If a TRS is confluent and terminating, then the result t_ℓ of a *computation* of s, $s \rightarrow_{\mathcal{R}} t_0 \rightarrow \ldots \rightarrow_{\mathcal{R}} t_\ell$ where t_ℓ is in normal form, is well-defined and unique. Hence this subclass of TRSs forms a model of deterministic computation. Note that the model is independent on the evaluation strategy.

In this paper we are interested in the *runtime complexity* of such computations, following [13] we measure runtime as the number of rewrite steps in relation to input sizes, and disregard starting terms that do not correspond to function calls: The set of *basic terms* $\mathcal{T}_{\mathsf{b}}$ constitutes of all terms $f(\vec{v})$ where f is defined and the arguments $\vec{v}$ are values. The *(innermost) runtime complexity* of a terminating TRS $\mathcal{R}$ is defined as $\mathrm{rc}_{\mathcal{R}}^{(\mathsf{i})}(n) := \max\{\mathsf{dh}(t, \rightarrow) \mid t \text{ is a basic of size up to } n\}$. Here $\rightarrow$ denotes $\rightarrow_{\mathcal{R}}$ or $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ respectively, and the *derivation height* $\mathsf{dh}(t, \rightarrow)$ is the maximal length of a derivation starting in t. As we focus in paper on innermost rewriting, we often drop the qualification *innermost*, when referring to the runtime complexity of a TRS. No confusion will arise from this.

4 The Small Polynomial Path Order

We arrive at the definition of sPOP*. To precisely assess the complexity of a TRS, sPOP* allows recursive definitions only on a subset of defined symbols, the so called *recursive symbols*. Symbols that are not recursive are called *compositional*.

Let $\mathcal{R}$ be a TRS and fix a precedence $\succcurlyeq$ on the symbols of $\mathcal{R}$. To assert our understanding that $\succcurlyeq$ reflects a call graph indifferent on equivalent symbols, we require that $\succcurlyeq$ is *admissible*: (i) constructors do not depend on defined symbols: $f \succ g$ implies that f is not a constructor, and (ii) the equivalence $\sim$ adheres the separation of constructors, recursive and compositional symbols: if $f \sim g$ then both f and g are either constructors, recursive or compositional symbols. The *depth of recursion* $\mathsf{rd}(f)$ is defined in correspondence to the rank $\mathsf{rk}(f)$, but only takes recursive symbols into account:

$$\mathsf{rd}(f) := \begin{cases} 1 + \max\{\mathsf{rd}(g) \mid f \succ g\} & \text{if } f \text{ is recursive, and} \\ \max\{\mathsf{rd}(g) \mid f \succ g\} & \text{otherwise} \ . \end{cases}$$

sPOP* does not differentiate between equivalent terms in principle, however the equivalence need to respect the separation of normal and safe argument positions. We formalise this in the equivalence relation $\overset{\mathsf{s}}{\sim}$, where $s \overset{\mathsf{s}}{\sim} t$ holds if $s = t$ or $s = f(s_1, \ldots, s_k \,;s_{k+1}, \ldots, s_{k+l})$, $t = g(t_1, \ldots, t_k \,;t_{k+1}, \ldots, t_{k+l})$ where f is equivalent to g and $s_i \overset{\mathsf{s}}{\sim} t_{\pi(i)}$ for all argument positions $i = 1, \ldots, k + l$. Here π denotes a permutation on argument positions so that position $\pi(i)$ is normal if and only if the position i is normal. Let $s = f(s_1, \ldots, s_k \,;s_{k+1}, \ldots, s_{k+l})$ we define the relation $\rhd_{\mathsf{n}}$ so that $s \rhd_{\mathsf{n}} t$ holds if (i) $s_i \overset{\mathsf{s}}{\sim} t$ or $s_i \rhd_{\mathsf{n}} t$ for some argument s_i of s, and (ii) if f is defined then the argument position i is normal ($i \in \{1, \ldots, k\}$). In any case $s \rhd_{\mathsf{n}} t$ implies that t is equivalent to a subterm of s.

The following definition introduces small polynomial path orders $>_{\mathsf{spop}*}$.

Definition 1. *Let s and t be terms such that $s = f(s_1,\dots,s_k\,;s_{k+1},\dots,s_{k+l})$. Then $s >_{\mathsf{spop}*} t$ if one of the following alternatives holds.*

1. *$s_i \geqslant_{\mathsf{spop}*} t$ for some argument s_i of s.*
2. *f is a defined symbol, $t = g(t_1,\dots,t_m\,;t_{m+1},\dots,t_{m+n})$ such that g is below f in the precedence and the following conditions hold:*
   a) *$s \rhd_{\mathsf{n}} t_j$ for all normal arguments t_j of t;*
   b) *$s >_{\mathsf{spop}*} t_j$ for all safe arguments t_j of t;*
   c) *at most one argument t_j of t contains defined symbols not below f in the precedence.*
3. *f is recursive and $t = g(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$ such that g is equivalent to f in the precedence and the following conditions hold:*
   a) *$\langle s_1,\dots,s_k\rangle >_{\mathsf{spop}*} \langle t_{\pi(1)},\dots,t_{\pi(k)}\rangle$ for some permutation π on normal argument positions;*
   b) *$\langle s_{k+1},\dots,s_{k+l}\rangle >_{\mathsf{spop}*} \langle t_{\tau(k+1)},\dots,t_{\tau(k+l)}\rangle$ for some permutation τ on normal argument positions.*

Here $s \geqslant_{\mathsf{spop}*} t$ denotes that either s and t are equivalent or $s >_{\mathsf{spop}*} t$ holds. In the last clause we use $>_{\mathsf{spop}*}$ also for the product extension of $>_{\mathsf{spop}*}$, where $\langle s_1,\dots,s_n\rangle \geqslant_{\mathsf{spop}*} \langle t_1,\dots,t_n\rangle$ means $s_i \geqslant_{\mathsf{spop}*} t_i$ for all $i = 1,\dots,n$, and $\langle s_1,\dots,s_n\rangle >_{\mathsf{spop}*} \langle t_1,\dots,t_n\rangle$ indicates that additionally $s_{i_0} >_{\mathsf{spop}*} t_{i_0}$ holds for at least one $i_0 \in \{1,\dots,n\}$.

We say that a TRS $\mathcal{R}$ is *compatible* with $>_{\mathsf{spop}*}$ if all rules are *oriented* from left to right: $l >_{\mathsf{spop}*} r$ for all rules $l \to r \in \mathcal{R}$. We use the notation $>^{\langle \mathsf{i}\rangle}_{\mathsf{spop}*}$ to refer to the i^{th} case in Definition 1 (a similar notation is employed for the subsequently defined orders).

Some Comments on the Definition Consider a compatible TRS $\mathcal{R}$. By compatibility the left-hand side l is compared to the right-hand side of r and, recursively the arguments of r, for each rule $l \to r$ of $\mathcal{R}$. The case $>^{\langle 1\rangle}_{\mathsf{spop}*}$ is standard in recursive orders and allows the treatment of functions defined by projection. Consider the more involved cases where the orientation is due to $>^{\langle 2\rangle}_{\mathsf{spop}*}$ or $>^{\langle 3\rangle}_{\mathsf{spop}*}$. We use case $>^{\langle 2\rangle}_{\mathsf{spop}*}$ to capture function composition, where f is defined in terms of a function g below f in the precedence. The order constraints on normal arguments enforce that safe arguments of f cannot pass to normal arguments of g, and moreover the use of $\rhd_{\mathsf{n}}$ disallows composition in normal positions of g. In contrast, safe arguments of the right-hand side can be compared using the full power of $>_{\mathsf{spop}*}$. The only additional restriction imposed states that at most one recursive call can occur below the function symbol g. Finally the case $>^{\langle 3\rangle}_{\mathsf{spop}*}$ captures recursion and is employed when l is compared to the recursive call. Here we require that the product of arguments decrease, where we are careful not to mix normal and safe arguments. In addition we require that normal arguments, ie. the recursion parameters, decrease strictly between l and the recursive call in r.

We say a constructor TRS $\mathcal{R}$ is *predicative recursive of degree d* if $\mathcal{R}$ is compatible with an instance $>_{\mathsf{spop}*}$ and the maximal depth of recursion of a function symbol in $\mathcal{R}$ is d.

$$
\begin{array}{rlrl}
+(0\,;y) & \to y & \times(0,y\,;\,) & \to 0 \\
+(\mathsf{s}(\,;x)\,;y) & \to \mathsf{s}(+(x\,;y)) & \times(\mathsf{s}(\,;x),y\,;\,) & \to +(y\,;\times(x,y\,;\,)) \\
\mathsf{sq}(x\,;\,) & \to \times(x\,;x) & &
\end{array}
$$

Figure 3: Rewrite system $\mathcal{R}_{\mathsf{sq}}$

$$
\begin{aligned}
\mathsf{f}_{i+1}(x\,;\,) &\to \mathsf{g}_{i+1}(x,x\,;\,) \\
\mathsf{g}_{i+1}(\mathsf{s}(\,;x),y\,;\,) &\to \mathsf{b}(\,;\mathsf{f}_i(y\,;\,),\mathsf{g}_{i+1}(x,y\,;\,))\ .
\end{aligned}
$$

Figure 4: Extension rules of TRS $\mathcal{R}_{i+1}$, where f_{i+1} and g_{i+1} are fresh

Theorem 2. *Let $\mathcal{R}$ be predicative recursive of degree d. Then the innermost derivation height of any basic term $f(\vec{u}\,;\vec{v})$ is bounded by a polynomial of degree d in the sum of the depths of normal arguments $\vec{u}$.*

As corollary to Theorem 2 we obtain that $>_{\mathsf{spop}*}$ induces polynomial innermost runtime complexity on constructor TRSs.

Corollary 3. *If $\mathcal{R}$ is predicative recursive of degree d, then the innermost runtime complexity of $\mathcal{R}$ lies in $O(n^d)$.*

Consider the constructor TRS $\mathcal{R}_{\mathsf{sq}}$, whose rules are given in Fig. 3. The TRS $\mathcal{R}_{\mathsf{sq}}$ defines squaring of natural numbers build from the constructors $\mathsf{0}$ and s. Consider the precedence so that $\mathsf{sq} > \times > + > \mathsf{s} \sim \mathsf{0}$. Then it can be verified that the TRS $\mathcal{R}_{\mathsf{sq}}$ is compatible with $>_{\mathsf{spop}*}$. For instance $\times(\mathsf{s}(\,;x),y\,;\,) >_{\mathsf{spop}*} +(y\,;\times(x,y\,;\,))$ follows by $>^{\langle 2\rangle}_{\mathsf{spop}*}$ since y appears as normal argument in the left-hand side and $\times(\mathsf{s}(\,;x),y\,;\,) >_{\mathsf{spop}*} \times(x,y\,;\,)$ follows by one application of $>^{\langle 3\rangle}_{\mathsf{spop}*}$. Note that the orientation only requires addition ($+$) and multiplication ($\times$) to be recursive symbols, but not the square function (sq). Hence the precedence gives a recursion depth of 2 for multiplication and squaring, and a recursion depth of 1 to addition. According to Theorem 2 addition gives rise to linear, and multiplication as well as squaring gives rise to quadratic runtime complexity. Overall, the runtime complexity is quadratic.

We emphasise that Corollary 3 is tight in the sense that for any $d \in \mathbb{N}$ there exists a predicative recursive TRS of degree d so that the runtime complexity is bounded from below by $\Omega(n^d)$.

To see this, define a family of TRSs $\mathcal{R}_i$ ($i \in \mathbb{N}$) inductively as follows: $\mathcal{R}_0 := \{\mathsf{f}_0(x\,;\,) \to \mathsf{a}\}$ and $\mathcal{R}_{i+1}$ extends $\mathcal{R}_i$ by the rules presented in Fig. 4. By construction $\mathcal{R}_d$ is compatible with $>_{\mathsf{spop}*}$ as induced by the precedence $\mathsf{f}_d > \mathsf{g}_d > \mathsf{f}_{d-1} > \mathsf{g}_{d-1} > \ldots > \mathsf{f}_0 > \mathsf{a} \sim \mathsf{b}$, where only the defined symbols g_i ($i = 1,\ldots,d$) are recursive. Obviously the maximal depth of recursion of $\mathcal{R}_d$ is d

We show that the runtime complexity of $\mathcal{R}_d$ is in $\Omega(n^d)$: For $d = 0$ this is immediate. For $d > 1$, note that g_d performs recursion on its first argument, at each step calling f_{d-1}. Conclusively $\mathsf{f}_d(\mathsf{s}^n(\mathsf{a}))$ calls n times the function f_{d-1}. Inductive reasoning yields that $\mathsf{f}_d(\mathsf{s}^n(\mathsf{a}))$ reduces in at least n^d steps.

5 Predicative Interpretations

In the following, let $\mathcal{R}$ denote a constructor TRS that is compatible with $>_{\mathsf{spop}*}$. To simplify matters, we suppose for now that $\mathcal{R}$ is also *completely defined.* Consider a rewrite rule $f(\vec{u}\,;\vec{v}) \to r \in \mathcal{R}$ that triggers an innermost rewrite step

$$s = C[f(\vec{u}\sigma\,;\vec{v}\sigma)] \xrightarrow{\mathsf{i}}_{\mathcal{R}} C[r\sigma] = t\ .$$

Since normal forms and values coincide, the rule is only triggered if all arguments $\vec{u}\sigma, \vec{v}\sigma$ of the redex are values. Due to the limitations imposed by $>^{\langle 2\rangle}_{\mathsf{spop}*}$ and $>^{\langle 3\rangle}_{\mathsf{spop}*}$, it is not difficult to see that if $r\sigma$ is not a value itself, then at least all normal arguments are values. We capture this observation in the set $\mathcal{T}^{\to}_{\mathsf{b}}$, defined as the least extension of values closed under $\mathcal{F}$ operations containing only values at normal argument positions: $\mathcal{T}^{\to}_{\mathsf{b}}$ is the least set such that (i) $\mathcal{V}\mathsf{al} \subseteq \mathcal{T}^{\to}_{\mathsf{b}}$, and (ii) if $f \in \mathcal{F}$, $\vec{s} \subseteq \mathcal{V}\mathsf{al}$ and $\vec{t} \subseteq \mathcal{T}^{\to}_{\mathsf{b}}$ then $f(\vec{s}\,;\vec{t}) \in \mathcal{T}^{\to}_{\mathsf{b}}$. This set is closed under rewriting.

Lemma 4. *Let $\mathcal{R}$ be a completely defined TRS compatible with $>_{\mathsf{spop}*}$. If $s \in \mathcal{T}^{\to}_{\mathsf{b}}$ and $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ then $t \in \mathcal{T}^{\to}_{\mathsf{b}}$.*

Proof. The Lemma follows by a straight forward inductive argument on Definition 1. ☐

Since $\mathcal{T}^{\to}_{\mathsf{b}}$ contains in particular all basic terms, it follows that the runtime complexity function $\mathrm{rc}^{\mathsf{i}}_{\mathcal{R}}$ depends only on terms from $\mathcal{T}^{\to}_{\mathsf{b}}$.

The *predicative interpretation* S maps terms from $\mathcal{T}^{\to}_{\mathsf{b}}$ to *sequences* of *normalised* terms by separating normal from safe arguments. Here a term is normalised if it is a term where arities of defined symbols correspond to the number of normal argument positions. We write f_{n} for the symbol f if it occurs in a normalised term. To denote sequences of terms, we use a fresh variadic function symbol $\circ$. Here variadic means that the arity of $\circ$ is finite but otherwise arbitrary. We always write $[a_1 \cdots a_n]$ for $\circ(a_1,\dots,a_n)$, in particular if we write $f(a_1,\dots,a_n)$ then $f \neq \circ$. We denote by $\mathcal{T}^*$ the set of *sequences* $[t_1 \cdots t_n]$ of normalised terms $t_1,\dots,t_n$. Abusing set-notation, we denote by $s \in [s_1 \cdots s_n]$ that $s = s_i$ for some $i \in \{1,\dots,n\}$.

The predicative interpretation S is defined on $\mathcal{T}^{\to}_{\mathsf{b}}$ as follows: If t is a value, then $\mathsf{S}(t) \coloneqq [\,]$. Otherwise if $t = f(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+l})$, then

$$\mathsf{S}(t) \coloneqq [\, f_{\mathsf{n}}(t_1,\dots,t_k)\,] \frown \mathsf{S}(t_{k+1}) \frown \cdots \frown \mathsf{S}(t_{k+l})\ .$$

Here the *concatenation* operator $\frown$ is defined on sequences such that $[s_1 \cdots s_n] \frown [t_1 \cdots t_n] \coloneqq [\,s_1 \cdots s_n\ t_1 \cdots t_m\,]$. We extend concatenation to terms by identifying terms t with the singleton sequences $[\,t\,]$, for instance $s \frown t = [\,s\ t\,]$.

In Fig. 5 we exemplify the predicative interpretation S on a rewrite step of the $\mathcal{R}_{\mathsf{sq}}$ depicted in Fig. 3.

6 The Small Polynomial Path Order on Sequences

We define the *small polynomial path order on sequences* $\mathcal{T}^*$. As these serve a purely technical reason, it suffices to represent the order via finite approximations $\blacktriangleright_k$. The parameter $k \in \mathbb{N}$ controls the width of terms and sequences. We lift terms equivalence to sequences by disregarding order of elements: $[s_1 \cdots s_n] \sim [t_1 \cdots t_n]$ if $s_i \sim t_{\pi(i)}$ for all $i = 1,\dots,n$ and some permutation π on $\{1,\dots,n\}$.

$$\begin{tikzcd}
\times(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)\,;\,) \arrow[r, "\mathsf{i}"] \arrow[d, "\mathsf{S}(\cdot)"] & {}_{\mathcal{R}_{\mathsf{sq}}} +(\mathsf{s}(0)\,;\times(\mathsf{s}(0),\mathsf{s}(0)\,;\,)) \arrow[d, "\mathsf{S}(\cdot)"] \\
{[\,\times_{\mathsf{n}}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0))\,]} & {[\,+_{\mathsf{n}}(\mathsf{s}(0))\ \ \times_{\mathsf{n}}(\mathsf{s}(0),\mathsf{s}(0))\,]}
\end{tikzcd}$$

Figure 5: Predicative interpretation

Definition 5. *Let $k \geqslant 1$, and let $\geqslant$ denote an admissible precedence. We define $\blacktriangleright_k$ inductively such that:*

1. *$f(s_1,\dots,s_n) \blacktriangleright_k g(t_1,\dots,t_m)$ if f is a defined symbol, g is below f in the precedence and the following conditions hold:*
   a) *all arguments t_j are equivalent to proper subterms of $f(s_1,\dots,s_n)$;*
   b) *$m \leqslant k$.*
2. *$f(s_1,\dots,s_n) \blacktriangleright_k g(t_1,\dots,t_n)$ if f is recursive and equivalent to g in the precedence and the following conditions hold:*
   a) *$\langle s_1,\dots,s_n\rangle \rhd\!/_{\!\sim} \langle t_{\pi(1)},\dots,t_{\pi(n)}\rangle$ for some permutation π;*
   b) *$n \leqslant k$.*
3. *$f(s_1,\dots,s_n) \blacktriangleright_k [t_1 \cdots t_m]$ and the following conditions hold:*
   a) *$f(s_1,\dots,s_n) \blacktriangleright_k t_j$ for all $j = 1,\dots,m$;*
   b) *at most one element t_j ($j \in \{1,\dots,m\}$) contains defined symbols not below f in the precedence;*
   c) *$m \leqslant k$.*
4. *$[s_1 \cdots s_n] \blacktriangleright_k b$ where b is equivalent to $b_1 \frown \cdots \frown b_n$ and the following conditions hold:*
   a) *$s_i \underline{\blacktriangleright}_k b_i$ for all $i = 1,\dots,n$;*
   b) *$s_{i_0} \blacktriangleright_k b_{i_0}$ for at least one $i_0 \in \{1,\dots,n\}$.*

Here $a \underline{\blacktriangleright}_k b$ denotes that either a and b are equivalent or $a \blacktriangleright_k b$ holds, and $\langle s_1,\dots,s_k\rangle \rhd\!/_{\!\sim} \langle t_i,\dots,t_k\rangle$ denotes that the term t_i is equivalent to a subterm of s_i for all $i = 1,\dots,n$, and at least one t_{i_0} is equivalent to a proper subterm of s_{i_0} ($i_0 \in \{1,\dots,n\}$).

In Fig. 6 we demonstrate that predicative interpretation S as exemplified in Fig. 5 embed the corresponding rewrite step into $\blacktriangleright_2$. Here we abbreviate $s = \times_{\mathsf{n}}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0))$.

The orders $\blacktriangleright_k$ is defined so that following conditions are satisfied:

Lemma 6. *Let $k \geqslant 1$. We have*

1. *$\blacktriangleright_l \subseteq \blacktriangleright_k$ for all $l \leqslant k$,*
2. *$\sim \cdot \blacktriangleright_k \cdot \sim \subseteq \blacktriangleright_k$, and*
3. *if $a \blacktriangleright_k b$ then $a \frown c \blacktriangleright_k b \frown c$.*

$$\begin{array}{rlr}
\text{1:} & s \blacktriangleright_2 +_{\mathsf{n}}(\mathsf{s}(0)) & \text{by } \blacktriangleright_k^{\langle 1\rangle} \\
\text{2:} & s \blacktriangleright_2 \times_{\mathsf{n}}(\mathsf{s}(0),\mathsf{s}(0)) & \text{by } \blacktriangleright_k^{\langle 2\rangle} \\
\text{3:} & s \blacktriangleright_2 [\,+_{\mathsf{n}}(\mathsf{s}(0))\ \times_{\mathsf{n}}(\mathsf{s}(0),\mathsf{s}(0))\,] & \text{by } \blacktriangleright_k^{\langle 3\rangle}\text{, using 1 and 2} \\
\text{4:} & [\,s\,] \blacktriangleright_2 [\,+_{\mathsf{n}}(\mathsf{s}(0))\ \times_{\mathsf{n}}(\mathsf{s}(0),\mathsf{s}(0))\,] & \text{by } \blacktriangleright_k^{\langle 4\rangle}\text{, using 3}
\end{array}$$

Figure 6: Predicative interpretation revisited

Proof. We focus on Property 3, the other facts either follow by definition, or by a straight forward inductive argument. In proof of Property 3 we perform case analysis on the last rule that concludes $a \blacktriangleright_k b$.

Suppose $a \blacktriangleright_k^{\langle 1\rangle} b$ or $a \blacktriangleright_k^{\langle 2\rangle} b$. Then $a = f(s_1,\dots,s_n)$ and $b = g(t_1,\dots,t_m)$. Hence $a \frown c = [\,f(s_1,\dots,s_n)\ u_1\ \cdots\ u_l\,]$ $b \frown c = g(t_1,\dots,t_m) \frown u_1 \frown \cdots \frown u_l$, where either c is the sequence $[u_1\ \cdots\ u_l]$ or c is the term u_1 and $l = 1$. We conclude the lemma using $\blacktriangleright_k^{\langle 4\rangle}$, employing the inequalities $a \blacktriangleright_k b$ and $u_i \mathrel{\underline{\blacktriangleright}}_k u_i$ for all $i \in \{1,\dots,l\}$.

Suppose $a \blacktriangleright_k^{\langle 3\rangle} b$. Then $a = f(s_1,\dots,s_n)$ and $b = [t_1\ \cdots\ t_m]$. We conclude $a \frown c \blacktriangleright_k b \frown c$ similar to above.

Suppose $a \blacktriangleright_k^{\langle 4\rangle} b$. In this case $a = [s_1\ \cdots\ s_n]$ and b is equivalent to $b_1 \frown \cdots \frown b_n$ such that $s_i \mathrel{\underline{\blacktriangleright}}_k b_i$ for all $i = 1,\dots,n$ and $s_{i_0} \blacktriangleright_k b_{i_0}$ for at least one $i_0 \in \{1,\dots,n\}$. Then $a \frown c = [\,s_1\ \cdots\ s_n\ u_1\ \cdots\ u_l\,]$ and $b \frown c$ is equivalent to $b_1 \frown \cdots \frown b_n \frown u_1 \frown \cdots \frown u_l$. One application of $\blacktriangleright_k^{\langle 4\rangle}$ proves the lemma. □

The length of $\blacktriangleright_k$ descending sequences is expressed by the function G_k, given by

$$\mathsf{G}_k(a) := 1 + \max\{\mathsf{G}_k(b) \mid a \blacktriangleright_k b\}\ .$$

Here a ranges over (normalised) terms and sequences. As intermediate result we obtain that sequences act purely as containers with respect to G_k.

Lemma 7. *Let $a = [t_1\ \cdots\ t_n]$ be sequence. Then $\mathsf{G}_k(a) = \sum_{i=1}^n \mathsf{G}_k(t_i)$.*

Proof. Let $a = [s_1\ \cdots\ s_n]$ be a sequence and observe $\mathsf{G}_k(a_1 \frown a_2) \geqslant \mathsf{G}_k(a_1) + \mathsf{G}_k(a_2)$. This is a consequence of Lemma 6 (3). Hence in particular $\mathsf{G}_k(a) = \mathsf{G}_k(s_1 \frown \cdots \frown s_n) \geqslant \sum_{i=1}^n \mathsf{G}_k(s_i)$ follows.

To prove the inverse direction, we show that $a \blacktriangleright_k b$ implies $\mathsf{G}_k(b) < \sum_{i=1}^n \mathsf{G}_k(s_i)$ by induction on $\mathsf{G}_k(a)$. The base case $\mathsf{G}_k(a) = 0$ follows trivially as the assumption $a \blacktriangleright_k b$ is not satisfied. For the inductive step, observe that $a \blacktriangleright_k b$ follows due to $\blacktriangleright_k^{\langle 4\rangle}$. Hence b is equivalent to $b_1 \frown \cdots \frown b_n$ where $s_i \mathrel{\underline{\blacktriangleright}}_k b_i$ for all $i = 1,\dots,n$ and $s_{i_0} \blacktriangleright_k b_{i_0}$ for at least one $i_0 \in \{1,\dots,n\}$. In particular, $\mathsf{G}_k(b_i) \leqslant \mathsf{G}_k(s_i)$ and $\mathsf{G}_k(b_{i_0}) < \mathsf{G}_k(s_{i_0})$. As we have $\mathsf{G}_k(b_i) \leqslant \mathsf{G}_k(b) < \mathsf{G}_k(a)$ for all $i \in \{1,\dots,n\}$, induction hypothesis is applicable to b and all b_i ($i \in \{1,\dots,n\}$). It follows that

$$\mathsf{G}_k(b) = \sum_{t \in b} \mathsf{G}_k(t) = \sum_{i=1}^n \sum_{t \in b_i} t = \sum_{i=1}^n \mathsf{G}_k(b_i) < \sum_{i=1}^n \mathsf{G}_k(s_i)\ .$$

□

Let r and d be natural number. We recursively define:

$$\mathsf{c}(r,d) := \begin{cases} 1 & \text{if } r = 1, \text{ and} \\ \mathsf{c}(r-1,d) \cdot k^{d+1} + 1 & \text{otherwise .} \end{cases}$$

Below the argument r will be instantiated by the rank and d by the depth of recursion of a function symbol f.

The next lemma is a technical lemma to ease the presentation of the proof of Theorem 9.

Lemma 8.

1. *Suppose $f(s_1,\dots,s_n) \blacktriangleright_k g(t_1,\dots,t_m)$ such that g is below f in the precedence. Further suppose*
$$\mathsf{G}_k(g(t_1,\dots,t_m)) \leqslant \mathsf{c}(\mathsf{rk}(g),\mathsf{rd}(g)) \cdot (2 + \sum_{i=1}^{m} \mathsf{dp}(t_i))^{\mathsf{rd}(g)} .$$
Then also $\mathsf{G}_k(g(t_1,\dots,t_m)) \leqslant \mathsf{c}(r-1,d) \cdot k^d \cdot (2+u)^{\mathsf{rd}(g)}$ where $u := \sum_{i=1}^{n} \mathsf{dp}(s_i)$, $r := \mathsf{rk}(f)$ and $d := \mathsf{rd}(f)$.

2. *Suppose $f(s_1,\dots,s_n) \blacktriangleright_k g(t_1,\dots,t_m)$ such that f is equivalent to g in the precedence. Further suppose that $\sum_{i=1}^{m} \mathsf{dp}(t_i) < \sum_{i=1}^{n} \mathsf{dp}(s_i)$ implies*
$$\mathsf{G}_k(g(t_1,\dots,t_m)) \leqslant \mathsf{c}(\mathsf{rk}(g),\mathsf{rd}(g)) \cdot (2 + \sum_{i=1}^{m} \mathsf{dp}(t_i))^{\mathsf{rd}(g)}$$
Then also $\mathsf{G}_k(g(t_1,\dots,t_m)) \leqslant \mathsf{c}(r,d) \cdot (1+u)^d$ where $u := \sum_{i=1}^{n} \mathsf{dp}(s_i)$, $r := \mathsf{rk}(f)$ and $d := \mathsf{rd}(f)$.

Proof. We consider the first point of the proposition. By assumption $f(s_1,\dots,s_n) \blacktriangleright_k^{\langle 1 \rangle} g(t_1,\dots,t_m)$. Consider a term t_j where $j \in \{1,\dots,m\}$. By definition, t_j is equivalent to a proper subterms of $f(s_1,\dots,s_n)$ and it follows that $\mathsf{dp}(t_j) \leqslant \sum_{i=1}^{n} \mathsf{dp}(s_i) =: u$. Hence $\sum_{i=1}^{m} \mathsf{dp}(t_i) \leqslant k \cdot u$ since $m \leqslant k$ by definition. As g is below f in the precedence, we have $\mathsf{rk}(g) < r$ and $\mathsf{rd}(g) \leqslant d$. We conclude by monotonicity of c and assumption

$$\begin{aligned} \mathsf{G}_k(g(t_1,\dots,t_m)) &\leqslant \mathsf{c}(\mathsf{rk}(g),\mathsf{rd}(g)) \cdot (2 + k \cdot u)^{\mathsf{rd}(g)} \\ &\leqslant \mathsf{c}(r-1,d) \cdot k^d \cdot (2+u)^{\mathsf{rd}(g)} . \end{aligned}$$

For the second point of the proposition, observe that by assumption $f(s_1,\dots,s_n) \blacktriangleright_k^{\langle 2 \rangle} g(t_1,\dots,t_m)$. Hence $m = n$ and the order constraints on arguments give $\sum_{i=1}^{n} \mathsf{dp}(t_i) < \sum_{i=1}^{n} \mathsf{dp}(s_i) =: u$. Using $\mathsf{rk}(g) = r$ and $\mathsf{rd}(g) = d$, the assumptions yield:

$$\begin{aligned} \mathsf{G}_k(g(t_1,\dots,t_n)) &\leqslant \mathsf{c}(\mathsf{rk}(g),\mathsf{rd}(g)) \cdot (2 + \sum_{i=1}^{m} \mathsf{dp}(t_i))^{\mathsf{rd}(g)} \\ &\leqslant \mathsf{c}(r,d) \cdot (1+u)^d . \end{aligned}$$

□

Theorem 9. *Let f be a defined symbol of recursion depth d. Then $\mathsf{G}_k(f(s_1,\dots,s_n)) \leqslant c \cdot \left(\sum_{i=1}^{n} \mathsf{dp}(s_i)\right)^d$ for all values $s_1,\dots,s_n$. Here the constant $c \in \mathbb{N}$ depends only on f and k.*

Proof. Let k be fixed. To show the theorem, we show for all terms $f(s_1, \dots, s_n)$, whose arguments are constructor terms, that $f(s_1, \dots, s_n) \blacktriangleright_k b$ implies:

$$\mathsf{G}_k(b) < \mathsf{c}(\mathsf{rk}(f), \mathsf{rd}(f)) \cdot \big(2 + \sum_{i=1}^{n} \mathsf{dp}(s_i)\big)^{\mathsf{rd}(f)} .$$

In proof we employ induction on $\mathsf{rk}(f)$ and side induction on $\sum_{i=1}^{n} \mathsf{dp}(s_i)$.

Consider $g(t_1, \dots, t_m)$ with $f \geqslant g$. We state the induction hypothesis (IH) and side induction hypothesis (SIH). IH states that if g is below f in the precedence then

$$\mathsf{G}_k(g(t_1, \dots, t_m)) \leqslant \mathsf{c}(\mathsf{rk}(g), \mathsf{rd}(g)) \cdot (2 + \sum_{i=1}^{m} \mathsf{dp}(t_i))^{\mathsf{rd}(g)} ,$$

while SIH states that if f and g are equivalent in the precedence but $\sum_{i=1}^{m} \mathsf{dp}(t_i) < \sum_{i=1}^{n} \mathsf{dp}(s_i)$ then

$$\mathsf{G}_k(g(t_1, \dots, t_m)) \leqslant \mathsf{c}(\mathsf{rk}(g), \mathsf{rd}(g)) \cdot (2 + \sum_{i=1}^{m} \mathsf{dp}(t_i))^{\mathsf{rd}(g)} .$$

Set $u := \sum_{i=1}^{n} \mathsf{dp}(s_i)$, $r := \mathsf{rk}(f)$ and $d := \mathsf{rd}(f)$ and assume $f(s_1, \dots, s_n) \blacktriangleright_k b$. We prove $\mathsf{G}_k(b) < \mathsf{c}(r, d) \cdot (2 + u)^d$.

In the base case $r = 1$ of the main induction, either $f(s_1, \dots, s_n) \blacktriangleright_k^{\langle 2 \rangle} b$ or $f(s_1, \dots, s_n) \blacktriangleright_k^{\langle 3 \rangle} b$ as f is minimal in the precedence. For the base case $u = 0$ of the side induction we see that $b = [\,]$, hence $\mathsf{G}_k(b) = 0$ and the theorem follows. For the inductive step of the side induction let $u > 0$. If f is compositional, the assumptions give $f(s_1, \dots, s_n) \blacktriangleright_k^{\langle 3 \rangle} b$, hence b is a sequence, in particular $b = [\,]$ and the theorem follows. For the remaining case that f is recursive, we consider two sub-cases: (i) b is a term, and (ii) b is a sequence. In the former sub-case, $f(s_1, \dots, s_n) \blacktriangleright_k^{\langle 2 \rangle} b$ where $b = g(t_1, \dots, t_n)$ and g is equivalent to f in the precedence. The order constraints on arguments give $\sum_{i=1}^{n} \mathsf{dp}(t_i) < u$. Employing $\mathsf{rd}(g) = d$ and $\mathsf{rk}(g) = r$, we conclude the sub-case by SIH:

$$\mathsf{G}_k(b) \leqslant \mathsf{c}(r, d) \cdot (2 + \sum_{i=1}^{n} \mathsf{dp}(t_i))^d < \mathsf{c}(r, d) \cdot (2 + u)^d .$$

In the second sub-case b is a sequence $[t_1 \cdots t_m]$ where $f(s_1, \dots, s_n) \blacktriangleright_k^{\langle 3 \rangle} b$. In particular, minimality of f in the precedence gives $m \leqslant 1$. Then the theorem follows trivially for $m = 0$, for $m = 1$ we conclude by the sub-case (i), additionally employing Lemma 7.

Now, we consider the step case of the main induction, let $r > 2$. If $b = g(t_1, \dots, t_m)$, then obviously $f \geqslant g$ and we conclude by Lemma 8. Note that IH (SIH) yield the assumptions of the lemmas.

On the other hand suppose $b = [t_1 \cdots t_m]$ and thus $f(a_1, \dots, a_n) \blacktriangleright_k b$ follows by $\blacktriangleright_k^{\langle 3 \rangle}$. Then $m \leqslant k$ and $f(a_1, \dots, a_n) \blacktriangleright_k t_j$ for all $j = 1, \dots, m$. Additionally at most one t_{j_0} ($j_0 \in \{1, \dots, m\}$) contains defined symbols not below f in the precedence. We analyse two sub-cases: either (i) f is recursive or (ii) f is compositional. We consider the first sub-case. In this case, $f > g$ implies $d > \mathsf{rd}(g) \geqslant 0$. Lemma 8 yields:

$$\begin{aligned} \mathsf{G}_k(t_j) &\leqslant \mathsf{c}(r-1, d) \cdot k^d \cdot (2 + u)^{d-1} && (\dagger) \\ \mathsf{G}_k(t_{j_0}) &\leqslant \mathsf{c}(r-1, d) \cdot k^d \cdot (2 + u)^{d-1} + \mathsf{c}(r, d) \cdot (1 + u)^d . \end{aligned}$$

Here (†) holds if $j \neq j_0$. Recall $m \leqslant k$ and $\mathsf{c}(r-1,d) \cdot k^{d+1} < \mathsf{c}(r,d)$. We conclude with Lemma 7:

$$\begin{aligned}
\mathsf{G}_k(b) &= \mathsf{G}_k(t_{j_0}) + \sum_{j=1, j \neq j_0}^{m} \mathsf{G}_k(t_j) \\
&\leqslant k \cdot (\mathsf{c}(r-1,d) \cdot k^d \cdot (2+u)^{d-1}) + \mathsf{c}(r,d) \cdot (1+u)^d \\
&< \mathsf{c}(r,d) \cdot (2+u)^{d-1} + \mathsf{c}(r,d) \cdot (1+u)^d \\
&\leqslant \mathsf{c}(r,d) \cdot (2+u)^d \; .
\end{aligned}$$

On the other hand, consider the second sub-case that f is compositional. Conclusively $f(a_1,\ldots,a_n) \blacktriangleright_k t_j$ can be strengthened to $f(a_1,\ldots,a_n) \blacktriangleright_k^{\langle 1 \rangle} t_j$. Employing that $d \geqslant \mathsf{rd}(g)$ for all symbols g below f in the precedence, using Lemma 7 and Lemma 8 we see

$$\begin{aligned}
\mathsf{G}_k(b) = \sum_{j=1}^{m} \mathsf{G}_k(b_j) &\leqslant k \cdot (\mathsf{c}(r-1,d) \cdot k^d \cdot (2+u)^d) \\
&< \mathsf{c}(r,d) \cdot (2+u)^d \; .
\end{aligned}$$

The theorem follows. □

7 Predicative Embedding

In this section we prove that for some constant $k \in \mathbb{N}$ depending only on the considered TRS $\mathcal{R}$, each innermost rewrite step gives rise to a $\blacktriangleright_k$ descent under predicative interpretation S. To ease the presentation, we provide the following auxiliary lemma.

Lemma 10. *Let $s = f(s_1,\ldots,s_l\,;\vec{s})$ be a basic term, let t be a term of size up to $\ell \in \mathbb{N}$ and let σ be a substitution that maps variables to values. If $s >_{\mathsf{spop}*} t$ then (i) $f_{\mathsf{n}}(s_1\sigma,\ldots,s_l\sigma) \blacktriangleright_\ell u$ for all $u \in \mathsf{S}(t\sigma)$ and further, (ii) at most one $u \in \mathsf{S}(t\sigma)$ contains a defined symbols not below f in the precedence.*

Proof. We prove Property (i) by induction on $>_{\mathsf{spop}*}$ and case analysis on the last rule concluding $s >_{\mathsf{spop}*} t$. Consider the case $s >_{\mathsf{spop}*}^{\langle 1 \rangle} t$ where $s_i \geqslant_{\mathsf{spop}*} t$ holds for some argument s_i of s. Since only case $>_{\mathsf{spop}*}^{\langle 1 \rangle}$ applies on values, we see that t is equivalent to subterm of s_i, in particular $t\sigma$ is a value and $\mathsf{S}(t\sigma) = [\,]$. Trivially the lemma follows.

Next, consider the case $s >_{\mathsf{spop}*}^{\langle 2 \rangle} t$. Then $t = g(t_1,\ldots,t_k\,;t_{k+1},\ldots,t_{k+n})$. Abbreviate $a = \mathsf{S}(t_{k+1}\sigma) \frown \cdots \frown \mathsf{S}(t_{k+n}\sigma)$ and hence by definition $\mathsf{S}(t\sigma) = [\,g_{\mathsf{n}}(t_1\sigma,\ldots,t_k\sigma)\,] \frown a$. As $s >_{\mathsf{spop}*} t_j$ for all safe arguments t_j of t, induction hypothesis gives $f_{\mathsf{n}}(s_1\sigma,\ldots,s_l\sigma) \blacktriangleright_k u$ for all $u \in a$. We verify

$$f_{\mathsf{n}}(s_1\sigma,\ldots,s_l\sigma) \blacktriangleright_\ell g_{\mathsf{n}}(t_1\sigma,\ldots,t_k\sigma) \; . \qquad (‡)$$

By the ordering constraints imposed by $>_{\mathsf{spop}*}^{\langle 2 \rangle}$, the defined symbol g is below the defined symbol f in the precedence, and $s \rhd_{\mathsf{n}} t_j$ for all normal arguments t_j of t. The latter reveals that the instances $t_j\sigma$ are equivalent to proper subterms of the normalised term $f_{\mathsf{n}}(s_1\sigma,\ldots,s_l\sigma)$. As trivially k is bounded by the size of t, one application of $\blacktriangleright_k^{\langle 1 \rangle}$ concludes (‡).

Finally, consider the case $s >^{\langle 3 \rangle}_{\mathsf{spop}*} t$. Then $t = g(t_1, \dots, t_l \,; t_{l+1}, \dots, t_{l+n})$ where g is equivalent to f in the precedence. By reasoning similar to the case $>^{\langle 1 \rangle}_{\mathsf{spop}*}$, the ordering constraint $\langle s_1, \dots, s_k \rangle >_{\mathsf{spop}*} \langle t_{\pi(1)}, \dots, t_{\pi(k)} \rangle$ on normal arguments reveals $\langle s_1\sigma, \dots, s_k\sigma \rangle \rhd\!\!\!/_{\sim} \langle t_{\pi(1)}\sigma, \dots, t_{\pi(k)}\sigma \rangle$. As l is trivially bounded by the size of t we conclude (‡) by $\blacktriangleright^{\langle 2 \rangle}_{k}$. Using the ordering constraints on safe arguments of t, we see that all safe arguments t_j of t are values, in particular $\mathsf{S}(t_j\sigma) = [\,]$. The lemma follows.

For Property (ii) a straight forward induction reveals that t contains at most one defined symbol not below f in the precedence. Here we make essential use of Condition (iii) in $>^{\langle 2 \rangle}_{\mathsf{spop}*}$. Then we conclude by the shape of σ and definition of predicative interpretations. □

Lemma 11. *Let $\mathcal{R}$ be a completely defined TRS compatible with $>_{\mathsf{spop}*}$. Let denote the maximal size of a right-hand side in $\mathcal{R}$. If $s \in \mathcal{T}^{\rightarrow}_{\mathsf{b}}$ and $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ then $\mathsf{S}(s) \blacktriangleright_{\ell} \mathsf{S}(t)$.*

Proof. Let $s \in \mathcal{T}^{\rightarrow}_{\mathsf{b}}$ and consider an innermost rewrite step $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$. We prove the lemma by induction on the rewrite context. In the base case, the context is empty, i.e., $s = l\sigma$ and $t = r\sigma$ for some rule $l \to r \in \mathcal{R}$ where $l = f(l_1, \dots, l_m \,; \vec{l})$ is a basic term. Since $\mathcal{R}$ is a completely defined TRS, all arguments of $l\sigma$ are values. Further compatibility gives $l >_{\mathsf{spop}*} r$, and hence all preconditions of Lemma 10 met. We conclude $f_{\mathsf{n}}(l_1\sigma, \dots, l_m\sigma) \blacktriangleright_{\ell} u$ for all terms $u \in \mathsf{S}(r\sigma)$, where at most one u is not constructed from defined symbols below f_{n} in the precedence. Exploiting that S erases values, hence in particular images of σ, it is not difficult to prove that the length of the sequence $\mathsf{S}(r\sigma)$ is bounded by $|r| \leqslant \ell$. In total we obtain $\mathsf{S}(s) = [\, f_{\mathsf{n}}(l_1\sigma, \dots, l_m\sigma) \,] \blacktriangleright_{\ell} \mathsf{S}(t)$ by one application of $\blacktriangleright^{\langle 3 \rangle}_{k}$ followed by one application of $\blacktriangleright^{\langle 4 \rangle}_{k}$.

Consider now the stepping case $s = f(\vec{v} \,; s_1, \dots, s_i, \dots, s_n)$ $t = f(\vec{v} \,; s_1, \dots, t_i, \dots, s_n)$ and $s_i \xrightarrow{\mathsf{i}}_{\mathcal{R}} t_i$. Here we use that since $s \in \mathcal{T}_{\mathsf{b}}$ all normal arguments $\vec{v}$ are values and cannot be rewritten. By induction hypothesis $\mathsf{S}(s_i) \blacktriangleright_{\ell} \mathsf{S}(t_i)$. Using Lemma 6 we see

$$
\begin{aligned}
\mathsf{S}(s\sigma) &= f_{\mathsf{n}}(\vec{v}) \frown \mathsf{S}(s_1) \frown \cdots \frown \mathsf{S}(s_i) \frown \cdots \frown \mathsf{S}(s_n) \\
&\blacktriangleright_{\ell} f_{\mathsf{n}}(\vec{v}) \frown \mathsf{S}(s_1) \frown \cdots \frown \mathsf{S}(t_i) \frown \cdots \frown \mathsf{S}(s_n) = \mathsf{S}(t\sigma) \;.
\end{aligned}
$$

□

Putting things together, we arrive at the proof of the main theorem.

Proof. Let $\mathcal{R}$ denote a predicative recursive TRS of degree d. We prove the existence of constant $c \in \mathbb{N}$ such that for all values $\vec{u}, \vec{v}$, the derivation height of the start term $f(\vec{u} \,; \vec{v})$ with respect to $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ is bounded by $c{\cdot}n^k$ where n is the sum of the depths of normal arguments $\vec{u}$.

Without loss of generality we can assume that $\mathcal{R}$ is completely defined. Otherwise we add sufficient rules to $\mathcal{R}$ that make it completely defined. For this we suppose, without loss of generality, that the signature $\mathcal{F}$ contains a constructor $\bot$ such that $f \geqslant \bot$ for all $f \in \mathcal{D}$. Note that if we add such a symbol, then still the precedence underlying $>_{\mathsf{spop}*}$ is admissible and the depth of recursion d does not increase. Call a normal form s *garbage* the root of s is defined. We extend $\mathcal{R}$ by adding the *garbage rules* rules $s \to \bot$ for all normal forms s which are garbage. Clearly $s >_{\mathsf{spop}*} \bot$ by one application of $>^{\langle 2 \rangle}_{\mathsf{spop}*}$, conclusively $\mathcal{R}_{\bot}$ is compatible with $>_{\mathsf{spop}*}$ too. Denote by $s_{\bot}$ the result of replacing garbage in a term s by $\bot$, that is $s_{\bot}$ is the unique normal form of s with respect to the garbage rules. Since garbage rules do not overlap with the constructor TRS $\mathcal{R}$, a straight forward inductive argument reveals that if $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ then also $s_{\perp} \xrightarrow{\mathsf{i}}{}^{+}_{\mathcal{R}_{\perp}} t_{\perp}$. Conclusively it suffices to estimate the number of rewrite steps induced by the completely defined TRS $\mathcal{R}_{\perp}$.

Set ℓ to denote the maximal size of a right-hand side of a rule in $\mathcal{R}_{\perp}$, and observe that ℓ is well defined. Consider a maximal derivation $f(\vec{u}\,;\vec{v}) \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\perp}} t_1 \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\perp}} \cdots \xrightarrow{\mathsf{i}}_{\mathcal{R}_{\perp}} t_n$. Let $i \in \{0, \ldots, n-1\}$. By Lemma 4 it follow that $t_i \in \mathcal{T}_{\mathsf{b}}^{\rightarrow}$, and consequently $\mathsf{S}(t_i) \blacktriangleright_{\ell} \mathsf{S}(t_{i+i})$ due to Lemma 11. So in particular the length n is bounded by the length of $\blacktriangleright_{\ell}$ descending sequences starting from $\mathsf{S}(f(\vec{u}\,;\vec{v})) = [\, f_{\mathsf{n}}(\vec{u})\,]$. Additionally using Lemma 7, Theorem 9 gives the constant $c \in \mathbb{N}$ as desired. □

8 Completeness Results

In this section we show that the small polynomial path order sPOP* is complete. Indeed we can even show something stronger. Let $\mathcal{R}$ be a TRS that makes only use of d nestings of safe recursion, then $\mathcal{R}$ is predicative recursive of degree d. Due to the weak form of safe composition we have the inclusion that $\mathcal{B}_{\mathsf{wsc}} \subseteq \mathcal{B}$. Concerning the converse inclusion, the following lemma states that the class $\mathcal{B}_{\mathsf{wsc}}$ is large enough to capture all the polytime computable functions.

Lemma 12. *Every polynomial time computable function belongs to* $\bigcup_{k \in \mathbb{N}} \mathcal{B}_{\mathsf{wsc}}^{k,0}$.

One can show this fact by following the proof of Theorem 3.7 in [12], where the unary variant of $\mathcal{B}_{\mathsf{wsc}}$ is defined and the inclusion corresponding to Lemma 12 is shown. We give the proof of Lemma 12 in detail below, but first assume the lemma in order to succinctly state our completeness result.

Theorem 13. *For any $\mathcal{B}_{\mathsf{wsc}}$-function f there exists a confluent TRS $\mathcal{R}_f$ that is predicative recursive of degree d, where d equals the maximal number of nested application of (*SRN*) in the definition of f.*

The completeness of sPOP* for the polytime computable functions is an immediate consequence of Lemma 12 and Theorem 13. The witnessing TRS $\mathcal{R}_f$ for $f \in \mathcal{B}_{\mathsf{wsc}}$ in Theorem 13 is obtained via a term rewriting characterisation of the class $\mathcal{B}_{\mathsf{wsc}}$. The term rewriting characterisation expresses the definition of $\mathcal{B}_{\mathsf{wsc}}$ as an *infinite* TRS $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ where the equations in Fig. 1 are oriented from left to right. Here binary words are formed from the constructor symbols ε, s_0 and s_1.

We define a one-to-one correspondence between the class $\mathcal{B}_{\mathsf{wsc}}$ of functions and the set of function symbols for $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ as follows. The function symbols $\mathsf{O}^{k,l}, \mathsf{I}_j^{k,l}, \mathsf{P}, \mathsf{C}$ correspond respectively to the initial functions $O^{k,l}, I_j^{k,l}, P, C$ of $\mathcal{B}_{\mathsf{wsc}}$. The symbol $\mathsf{SUB}[h, i_1, \ldots, i_k, \vec{g}]$ is used to denote the function obtained by composing functions h and $\vec{g}$ according to the schema of (WSC) Finally, the function symbol $\mathsf{SRN}[g, h_0, h_1]$ corresponds to the function defined by safe recursion on notation from g, h_0 and h_1 in accordance to the schema (SRN). It is easy to see that $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ is a constructor TRS. Further $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ is a orthogonal TRS, thus confluent.

Proof. Let f be an arbitrary function from $\mathcal{B}_{\mathsf{wsc}}$. By induction according to the definition of f in $\mathcal{B}_{\mathsf{wsc}}$ we show the existence of a TRS $\mathcal{R}_f$ and a precedence $\succcurlyeq_f$ such that

1. $\mathcal{R}_f$ is a finite restriction of $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$,
2. $\mathcal{R}_f$ contains the rule(s) that defines the function symbol f corresponding to f,
3. $\mathcal{R}_f$ is compatible with the sPOP*induced by $\succcurlyeq_f$,
4. f is maximal in the precedence $\succcurlyeq_f$, and
5. the maximal depth of recursion of the function symbols, i.e., $\mathsf{rd}(\mathsf{f})$, equals the maximal number of nested application of (SRN) in the definition of f in $\mathcal{B}_{\mathsf{wsc}}$.

To exemplify the construction we consider the step case that f is defined from some functions $g, h_0, h_1 \in \mathcal{B}_{\mathsf{wsc}}$ by the schema (SRN). By induction hypothesis we can find witnessing TRSs $\mathcal{R}_g, \mathcal{R}_{h_0}, \mathcal{R}_{h_1}$ and witnessing precedences $\succcurlyeq_g, \succcurlyeq_{h_0}, \succcurlyeq_{h_1}$ respectively for g, h_0, h_1. Extend the set of function symbols by the recursive symbol $\mathsf{f} :\equiv \mathsf{SRN}[\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1]$. Let $\mathcal{R}_f$ be the TRS consisting of $\mathcal{R}_g$, $\mathcal{R}_{h_0}$, $\mathcal{R}_{h_1}$ and the following three rules:

1. $\mathsf{f}(\varepsilon, \vec{x}; \vec{y}) \to \mathsf{g}(\vec{x}; \vec{y})$.
2. $\mathsf{f}(\mathsf{s}_i(; z), \vec{x}; \vec{y}) \to \mathsf{h}_i(z, \vec{x}; \vec{y}, \mathsf{f}(z, \vec{x}; \vec{y}))$ $(i = 0, 1)$.

Define the precedence $\succcurlyeq_f$ extending $\succcurlyeq_g \cup \succcurlyeq_{h_0} \cup \succcurlyeq_{h_1}$ by

– $\mathsf{f} \sim \mathsf{f}$ and

– $\mathsf{f} \succ \mathsf{g}'$ for any $\mathsf{g}' \in \{\mathsf{g}, \mathsf{h}_0, \mathsf{h}_1\}$.

Let $>_{\mathsf{spop}*}$ be the sPOP*induced by $\succcurlyeq_f$. Then it is easy to check that $\mathcal{R}_f$ enjoys Condition 1) and 2). In order to show Condition 3), it suffices to orient the three new rules by $>_{\mathsf{spop}*}$. For the rule in 1), $\mathsf{f}(\varepsilon, \vec{x}; \vec{y}) >^{\langle 2 \rangle}_{\mathsf{spop}*} \mathsf{g}(\vec{x}; \vec{y})$ holds by the definition of $\succcurlyeq_f$. For the remaining two rules in 2) we only orient the case $i = 0$. It is clear that $\mathsf{f}(\mathsf{s}_0(; z), \vec{x}; \vec{y}) >^{\langle 1 \rangle}_{\mathsf{spop}*} u$ holds for any u from $z, \vec{x}, \vec{y}$. In particular $\mathsf{s}_0(; z) >^{\langle 1 \rangle}_{\mathsf{spop}*} z$ holds. Hence $\mathsf{f}(\mathsf{s}_0(; z), \vec{x}; \vec{y}) >^{\langle 3 \rangle}_{\mathsf{spop}*} \mathsf{f}(z, \vec{x}; \vec{y})$ holds. This together with the definition of the precedence $\succcurlyeq_f$ allows us to conclude

$$\mathsf{f}(\mathsf{s}_0(; z), \vec{x}; \vec{y}) >^{\langle 2 \rangle}_{\mathsf{spop}*} \mathsf{h}_0(z, \vec{x}; \vec{y}, \mathsf{f}(z, \vec{x}; \vec{y})).$$

Consider Condition 4). For each $g' \in \{g, h_0, h_1\}$, g' is maximal in the precedence $\succcurlyeq_{g'}$ by induction hypothesis for g'. Hence by the definition of $\succcurlyeq_f$, f is maximal in $\succcurlyeq_f$. It remains to show Condition 5). Since f is a recursive symbol $\mathsf{rd}(\mathsf{f}) = 1 + \max\{\mathsf{rd}(\mathsf{g}), \mathsf{rd}(\mathsf{h}_0), \mathsf{rd}(\mathsf{h}_1)\}$. Without loss of generality let us suppose $\mathsf{rd}(\mathsf{g}) = \max\{\mathsf{rd}(\mathsf{g}), \mathsf{rd}(\mathsf{h}_0), \mathsf{rd}(\mathsf{h}_1)\}$. Then by induction hypothesis for g, $\mathsf{rd}(\mathsf{g})$ equals the maximal number of nested application of (SRN)in the definition of g in $\mathcal{B}_{\mathsf{wsc}}$. Hence $\mathsf{rd}(\mathsf{f}) = 1 + \mathsf{rd}(\mathsf{g})$ equals the one in the definition of f in $\mathcal{B}_{\mathsf{wsc}}$. □

In the sequel of this section, we provide the (technical) proof of Lemma 12. In what follows we totally follow presentations by W.G. Handley and S.S. Wainer in [12, Section 3]. We start with defining the k-th iteration $f^{(k)} \in \mathcal{B}_{\mathsf{wsc}}$ of $f \in \mathcal{B}_{\mathsf{wsc}}$ by

$$f^{(0)}(\vec{x}; \vec{y}, z) = z, \text{ and}$$
$$f^{(k+1)}(\vec{x}; \vec{y}, z) = f(\vec{x}; \vec{y}, f^{(k)}(\vec{x}; \vec{y}, z)) \; .$$

By the definition it is easy to see that $f^{(k)}(\vec{x}; \vec{y}, f^{(l)}(\vec{x}; \vec{y}, z)) = f^{(k+l)}(\vec{x}; \vec{y}, z)$ holds.

Lemma 14. *(Cf. [12, Lemma 3.3]) Let p be an n-ary polynomial with non-negative coefficients. If $f(x_1,\dots,x_n;\vec{y},z) \in \mathcal{B}_{\mathsf{wsc}}$, then there exists a function $\mathsf{iter}[p,f] \in \mathcal{B}_{\mathsf{wsc}}$ such that $\mathsf{iter}[p,f](\vec{x};\vec{y},z) = f^{(p(|\vec{x}|))}(\vec{x};\vec{y},z)$.*

Proof. We define a witnessing function $\mathsf{iter}[p,f]$ by induction over the construction of the polynomial p. In case that p is a trivial polynomial, i.e., $p(\vec{x}) = 0$ or $p(\vec{x}) = 1$, the choice of $\mathsf{iter}[p,f]$ is clear.

Case. $p(\vec{x}) = q(\vec{x}) + r(\vec{x})$ for some polynomials q,r: In this case $\mathsf{iter}[p,f]$ is defined by $\mathsf{iter}[p,f](\vec{x};\vec{y},z) = \mathsf{iter}[q,f](\vec{x};\vec{y},\mathsf{iter}[r,f](\vec{x};\vec{y},z))$.

Case. $p(\vec{x}) = x_j \cdot q(\vec{x})$ for some $j \in \{1,\dots n\}$ and for some polynomial $q(\vec{x})$: In this case we define an auxiliary function $\mathsf{loop}[q,f] \in \mathcal{B}_{\mathsf{wsc}}$ by

$$\begin{aligned}\mathsf{loop}[q,f](\epsilon,\vec{x};\vec{y},z) &= z,\\ \mathsf{loop}[q,f](ui,\vec{x};\vec{y},z) &= \mathsf{iter}[q,f](\vec{x};\vec{y},\mathsf{loop}[q,f](u,\vec{x};\vec{y},z)) \qquad (i=0,1)\end{aligned}$$

Then $\mathsf{loop}[p,f]$ is defined by $\mathsf{iter}[q,f](\vec{x};\vec{y},z) = \mathsf{loop}[q,f](x_j,\vec{x};\vec{y},z)$. One can check that $f^{(|u|\cdot q(|\vec{x}|))}(\vec{x};\vec{y},z) = \mathsf{loop}[q,f](u,\vec{x};\vec{y},z)$ holds by induction on $|u|$. □

We write $y \oplus x$ to denote the sequence y followed by $|x|$ 0's. Namely the operator $\oplus$ satisfies the equations $y \oplus \epsilon = y$ and $y \oplus (x0) = y \oplus (x1) = (y \oplus x)0$. We will write $y \oplus x \oplus x'$ instead of $(y \oplus x) \oplus x'$.

Lemma 15. *(Cf. [12, Lemma 3.4]) Let f,h be arbitrary functions. If f and h enjoy the condition*

$$\begin{aligned}&f(\vec{x},\epsilon,\vec{b},c,d) = c, \ \text{ and}\\ &f(\vec{x},a0,\vec{b},c,d) = f(\vec{x},a1,\vec{b},c,d) = h(\vec{x},d\oplus a,\vec{b},f(\vec{x},a,\vec{b},c,d)),\end{aligned}$$

then f also enjoys the condition

$$f(\vec{x},u,\vec{b},f(\vec{x},t,\vec{b},c,w),w\oplus t) = f(\vec{x},t\oplus u,\vec{b},c,w)\ .$$

Proof. By induction on $|u|$. For the base case $f(\vec{x},\epsilon,\vec{b},f(\vec{x},t,\vec{b},c,w),w\oplus t) = f(\vec{x},t,\vec{b},c,w) = f(\vec{x},t\oplus\epsilon,\vec{b},c,w)$. For the induction step

$$\begin{aligned}f(\vec{x},ui,\vec{b},f(\vec{x},t,\vec{b},c,w),w\oplus t) &= h(\vec{x},w\oplus t\oplus u,\vec{b},f(\vec{x},u,\vec{b},f(\vec{x},t,\vec{b},c,w),w\oplus t))\\ &= h(\vec{x},w\oplus t\oplus u,\vec{b},f(\vec{x},t\oplus u,\vec{b},c,w)) \qquad \text{(by IH)}\\ &= f(\vec{x},t\oplus ui,\vec{b},c,w).\end{aligned}$$

□

Let p be a polynomial with non-negative coefficients. Let us define two functions $\oplus_p$ and $\ominus_p$ by $\oplus_p(\vec{x};y) = \mathsf{iter}[p,S_0](\vec{x};y)$ and $\ominus_p(\vec{x};y) = \mathsf{iter}[p,P](\vec{x};y)$ for the predecessor function P. Then $\oplus_p,\ominus_p \in \mathcal{B}_{\mathsf{wsc}}$ by Lemma 14. By definition $\oplus_p(\vec{x};y)$ denotes the sequence y followed by $p(|\vec{x}|)$ 0's and $\ominus_p(\vec{x};y)$ denotes the sequence consisting of the first $|y| - p(|\vec{x}|)$ symbols of y (if $p(|\vec{x}|) \leqslant |y|$). In the following we use the operator "min" in a modified sense that $\min(x,y) = x$ if $|x| < |y|$, or otherwise $\min(x,y) = y$.

Lemma 16. *(Cf. [12, Lemma 3.5]) Let f and h enjoy the condition in the premise of Lemma 15. Let p be a polynomial p with non-negative coefficients. If there exists a function $h' \in \mathcal{B}_{\mathsf{wsc}}$ such that $h'(\vec{x}, a, \vec{b}, c) = h(\vec{x}, a, \vec{b}, c)$ holds for all a, c, then there exists a function $f[p] \in \mathcal{B}_{\mathsf{wsc}}$ such that $f[p](\vec{x}; a, \vec{b}, \vec{c}, d) = f(\vec{x}, \min(a, \oplus_p(\vec{x}; \epsilon)), \vec{b}, c, d)$ holds for all a, c and d.*

Proof. By induction over the construction of the polynomial p. In the special case that $p(\vec{x}) = 0$ the choice of the witnessing function $f[p]$ is clear.

Case. $p(\vec{x}) = 1$: In this case $f[p] \in \mathcal{B}_{\mathsf{wsc}}$ is defined by

$$f[p](\vec{x}; a, \vec{b}, c, d) = \begin{cases} c & \text{if } a = \epsilon, \\ h'(\vec{x}; d, \vec{b}, c) & \text{otherwise.} \end{cases}$$

Then the function $f[p]$ is as required since by the modified definition of min, $\min(a, 0) = \epsilon$ if $a = \epsilon$, or otherwise $\min(a, 0) = 0$.

Case. $p(\vec{x}) = q(\vec{x}) + r(\vec{x})$ for some polynomials q and r: In this case a function $f[p]$ is defined by

$$f[p](\vec{x}; a, \vec{b}, c, d) = f[q](\vec{x}; \ominus_r(\vec{x}; a), \vec{b}, f[r](\vec{x}; a, \vec{b}, c, d), \oplus_r(\vec{x}; d)) \ .$$

We show that $f[p](\vec{x}; a, \vec{b}, c, d) = f(\vec{x}, \min(a, \oplus_p(\vec{x}; \epsilon)), \vec{b}, c, d)$ holds by (sub)case-analysis.

Subcase 1. $|a| \leqslant r(|\vec{x}|)$: In this case $\ominus_r(\vec{x}; a) = \epsilon$. Hence the following equality holds:

$$\begin{aligned} f[p](\vec{x}; a, \vec{b}, c, d) &= f[q](\vec{x}; \epsilon, \vec{b}, f[r](\vec{x}; a, \vec{b}, c, d), \oplus_r(\vec{x}; d)) \\ &= f(\vec{x}, \min(0, \oplus_q(\vec{x}; \epsilon)), \vec{b}, f[r](\vec{x}; a, \vec{b}, c, d), \oplus_r(\vec{x}; d)) && \text{(by IH on } q) \\ &= f(\vec{x}, \epsilon, \vec{b}, f[r](\vec{x}; a, \vec{b}, c, d), \oplus_r(\vec{x}; d)) \\ &= f[r](\vec{x}; a, \vec{b}, c, d) && \text{(by Lemma 15)} \\ &= f(\vec{x}, \min(a, \oplus_r(\vec{x}; \epsilon), \vec{b}, c, d) && \text{(by IH on } r) \\ &= f(\vec{x}, \min(a, \oplus_p(\vec{x}; \epsilon), \vec{b}, c, d) \end{aligned}$$

Here the last equality follows as $\min(a, \oplus_r(\vec{x}; \epsilon)) = a = \min(a, \oplus_p(\vec{x}; \epsilon))$.

Subcase 2. $r(|\vec{x}|) < |a| \leqslant q(|\vec{x}|) + r(|\vec{x}|)$: In this case $|\ominus_r(\vec{x}; a)| = |a| - r(|\vec{x}|) > 0$ and further $\min(\ominus_r(\vec{x}; a), \oplus_q(\vec{x}; \epsilon)) = \ominus_r(\vec{x}; a)$ hold. Hence the following equality holds:

$$\begin{aligned} f[q](\vec{x}; a, \vec{b}, c, d) &= f[q](\vec{x}; \ominus_r(\vec{x}; a), \vec{b}, f[r](\vec{x}; a, \vec{b}, c, d), \oplus_r(\vec{x}; d)) \\ &= f(\vec{x}, \ominus_r(\vec{x}; a), \vec{b}, f(\vec{x}, \oplus_r(\vec{x}; \epsilon), \vec{b}, c, d), \oplus_r(\vec{x}; d)) && \text{(by IH on } q \text{ and } r) \\ &= f(\vec{x}, \oplus_r(\vec{x}; \epsilon) \oplus \ominus_r(\vec{x}; a), \vec{b}, c, d) && \text{(by Lemma 15)} \\ &= f(\vec{x}; a, \vec{b}, c, d) \ . \end{aligned}$$

The last equality holds since $| \oplus_r (\vec{x}; \epsilon) \oplus \ominus_r(\vec{x}; a)| = |a|$ and $f(\vec{x}, a, \vec{b}, c, d) = f(\vec{x}, a', \vec{b}, c, d)$ if $|a| = |a'|$ by the assumption on f. This allows us to conclude

$$f[q](\vec{x}; a, \vec{b}, c, d) = f(\vec{x}, \min(a, \oplus_p(\vec{x}; \epsilon)), \vec{b}, c, d) \ .$$

Subcase 3. $q(|\vec{x}|) + r(|\vec{x}|) < |a|$: In this case $| \ominus_r (\vec{x}; a)| = |a| - r(|\vec{x}|) > 0$ and further $\min(\ominus_r(\vec{x}; a), \oplus_q(\vec{x}; \epsilon)) = \ominus_q(\vec{x}; \epsilon)$. Hence the following equality holds:

$$\begin{aligned} f[p](\vec{x}; a, \vec{b}, c, d) &= f[q](\vec{x}; \ominus_r(\vec{x}; a), \vec{b}, f[r](\vec{x}; a, \vec{b}, c, d), \oplus_r(\vec{x}; d)) \\ &= f(\vec{x}, \oplus_q(\vec{x}; \epsilon), \vec{b}, f(\vec{x}, \oplus_r(\vec{x}; \epsilon)), \oplus_r(\vec{x}; d)) && \text{(by IH on } q \text{ and } r) \\ &= f(\vec{x}, \oplus_q(\vec{x}; \epsilon), \vec{b}, c, d) && \text{(by Lemma 15)} \\ &= f(\vec{x}, \min(a, \oplus_p(\vec{x}; \epsilon)), \vec{b}, c, d). \end{aligned}$$

This completes the case.

Case. $p(\vec{x}) = x_j \cdot q(\vec{x})$ for some j and for some polynomial q: In this case by IH there exists a witnessing function $f[q] \in \mathcal{B}_{\mathsf{wsc}}$ on q. Let us define a polynomial q' by $q'(z, \vec{x}) = q(\vec{x})$. Then a witnessing function $f[q'] \in \mathcal{B}_{\mathsf{wsc}}$ can be defined by $f[q'](z, \vec{x}; a, \vec{b}, c, d) = f[q](\vec{x}; a, \vec{b}, c, d)$ since $q'(z, \vec{x}) = q(\vec{x})$ holds. In order to define $f[p]$ we introduce an auxiliary polynomial p' by $p'(z, \vec{x}) = z \cdot q'(z, \vec{x})$. Further we define an auxiliary function $f^1_{p'} \in \mathcal{B}_{\mathsf{wsc}}$ via $f[q']$ by

$$\begin{aligned} &f^1_{p'}(\epsilon, \vec{x}; a, \vec{b}, c, d) = c, \\ &f^1_{p'}(zi, \vec{x}; a, \vec{b}, c, d) = f[q'](z, \vec{x}; \ominus_{p'}(z, \vec{x}; a), \vec{b}, f^1_{p'}(z, \vec{x}; a, \vec{b}, c, d), \oplus_{p'}(z, \vec{x}; a)). \quad (i = 0, 1) \end{aligned}$$

Now a function $f[p] \in \mathcal{B}_{\mathsf{wsc}}$ is defined by $f[p](\vec{x}; a, \vec{b}, c, d) = f^1_{p'}(x_j, \vec{x}; a, \vec{b}, c, d)$.

Claim 17. $f^1_{p'}(z, \vec{x}; a, \vec{b}, c, d) = f(\vec{x}; \min(a, \oplus_{p'}(z, \vec{x}; \epsilon)), \vec{b}, c, d)$.

Assuming the claim, we can conclude that

$$\begin{aligned} f[p](\vec{x}; a, \vec{b}, c, d) =& f^1_{p'}(x_j, \vec{x}; a, \vec{b}, c, d) \\ =& f(\vec{x}, \min(a, \oplus_{p'}(x_j, \vec{x}; \epsilon)), \vec{b}, c, d) \\ =& f(\vec{x}, \min(a, \oplus_p(\vec{x}; \epsilon)), \vec{b}, c, d) \ . \end{aligned}$$

We show the claim by (side) induction on $|z|$. In the base case

$$f^1_{p'}(\epsilon, \vec{x}; a, \vec{b}, c, d) = c = f(\vec{x}, \min(a, \oplus_{p'}(z, \vec{x}; \epsilon)), \vec{b}, c, d)$$

since $(a, \oplus_{p'}(z, \vec{x}; \epsilon)) = \epsilon$. In the induction case arguments split into three subcases.

Subcase 1. $|a| \leqslant p'(|z|, |\vec{x}|)$: In this case $\ominus_{p'}(z, \vec{x}; a) = \epsilon$ holds. Hence the following equality holds:

$$\begin{aligned} f^1_{p'}(zi, \vec{x}; a, \vec{b}, c, d) &= f[q'](z, \vec{x}; \ominus_{p'}(z, \vec{x}; a), \vec{b}, f^1_{p'}(z, \vec{x}; a, \vec{b}, c, d), \oplus_{p'}(z, \vec{x}; a)) \\ &= f(z, \vec{x}, \epsilon, \vec{b}, f^1_{p'}(z, \vec{x}; a, \vec{b}, c, d), \oplus_{p'}(z, \vec{x}; a)) && \text{(by IH on } q) \\ &= f^1_{p'}(z, \vec{x}; a, \vec{b}, c, d) \\ &= f(\vec{x}, \min(a, \oplus_{p'}(z, \vec{x}; a)), \vec{b}, c, d) && \text{(by SIH)} \\ &= f(\vec{x}, \min(a, \oplus_{p'}(zi, \vec{x}; a)), \vec{b}, c, d) \ . \end{aligned}$$

Subcase 2. $p'(|z|,|\vec{x}|) < |a| \leqslant p'(|z|+1,|\vec{x}|)$: In this case $|\ominus_{p'}(z,\vec{x};a)| = |a| - p'(|z|,|\vec{x}|)$ and $\min(\ominus_{p'}(z,\vec{x};a), \oplus_{p'}(z,\vec{x};a)) = \ominus_{p'}(z,\vec{x};a)$ hold. Hence the following equality holds:

$$\begin{aligned}
&f^1_{p'}(zi,\vec{x};a,\vec{b},c,d)\\
&\quad= f[q'](z,\vec{x};\ominus_{p'}(z,\vec{x};a),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f[q](\vec{x};\ominus_{p'}(z,\vec{x};a),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f(\vec{x},\min(\ominus_{p'}(z,\vec{x};a),\oplus_{p'}(z,\vec{x};a)),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a)) && \text{(by IH on } q\text{)}\\
&\quad= f(\vec{x},\ominus_{p'}(z,\vec{x};a),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f(\vec{x},\ominus_{p'}(z,\vec{x};a),\vec{b},f(\vec{x};\min(a,\oplus_{p'}(z,\vec{x};\epsilon)),\vec{b},c,d),\oplus_{p'}(z,\vec{x};a)) && \text{(by SIH)}\\
&\quad= f(\vec{x},\ominus_{p'}(z,\vec{x};a),\vec{b},f(\vec{x};\oplus_{p'}(z,\vec{x};\epsilon),\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f(\vec{x},a,\vec{b},c,d) && \text{(by Lemma 15)}\\
&\quad= f(\vec{x},\min(a,\oplus_{p'}(zi,\vec{x};\epsilon)),\vec{b},c,d)\ .
\end{aligned}$$

Subcase 3. $p'(|z|+1,|\vec{x}|) < |a|$: As in the previous subcase, the following equality holds:

$$\begin{aligned}
&f^1_{p'}(zi,\vec{x};a,\vec{b},c,d)\\
&\quad= f[q'](z,\vec{x};\ominus_{p'}(z,\vec{x};a),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f[q](\vec{x};\ominus_{p'}(z,\vec{x};a),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f(\vec{x},\min(\ominus_{p'}(z,\vec{x};a),\oplus_{p'}(z,\vec{x};a)),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a)) && \text{(by IH on } q\text{)}\\
&\quad= f(\vec{x},\oplus_{p'}(z,\vec{x};a),\vec{b},f^1_{p'}(z,\vec{x};a,\vec{b},c,d),\oplus_{p'}(z,\vec{x};a))\\
&\quad= f(\vec{x},\oplus_{p'}(z,\vec{x};a),\vec{b},f(\vec{x},\min(a,\oplus_{p'}(z,\vec{x};\epsilon)),\vec{b},c,d),\oplus_{p'}(z,\vec{x};a)) && \text{(by SIH)}\\
&\quad= f(\vec{x},\ominus_{p'}(z,\vec{x};a),\vec{b},f(\vec{x};\oplus_{p'}(z,\vec{x};\epsilon),\vec{b},c,d),\oplus_{p'}(z,\vec{x};a)w)\\
&\quad= f(\vec{x},\ominus_{p'}(zi,\vec{x};a),\vec{b},c,d) && \text{(by Lemma 15)}\\
&\quad= f(\vec{x},\min(a,\oplus_{p'}(zi,\vec{x};\epsilon)),\vec{b},c,d).
\end{aligned}$$

This completes the case and hence the proof of the lemma. □

Lemma 18 (Recursion simulation lemma)**.** *(Cf. [12, Theorem 3.6]) Let f be an l-ary polynomial-time function. Then for any polynomial $p : \mathbb{N}^k \to \mathbb{N}$ with non-negative coefficients there exists a function $\{f,p\} \in \mathcal{B}^{k,l}_{\mathsf{wsc}}$ such that for all $\vec{b}$, $\{f,p\}(\vec{x};\vec{b}) = f(\vec{b})$ holds whenever $\max|\vec{b}| \leqslant p(|\vec{x}|)$.*

Proof. We employ the recursion-theoretic characterisation of the polynomial time functions by A. Cobham [9]. Namely the class of all the polynomial-time computable functions coincides with the class $\mathcal{C}$, which is the smallest class containing the constant function $(x_1,\dots,x_k) \mapsto \varepsilon$, the successor functions $x \mapsto x0$ and $x \mapsto x1$, and the projection functions $(x_1,\dots,x_k) \mapsto x_j$, and closed under composition and polynomially length-bounded recursion on notation. Let f be a polynomial-time computable function. We show Recursion simulation lemma by induction over the construction of f in the Cobham class $\mathcal{C}$. If f is one of the initial functions, then the choice of the witnessing function $\{f,p\}$ is clear.

Case. f is defined by composition from some polynomial time functions $h, g_1,\dots,g_{l'}$ by $f(\vec{x}) = h(g_1(\vec{x}),\dots,g_{l'}(\vec{x}))$: Let an arbitrary polynomial p with non-negative coefficients be given. Then by IH for $g_1, \ldots, g_{l'}$ for each $j = 1, \ldots, l'$ there exists a witnessing function $\{g_j, p\} \in \mathcal{B}_{\mathsf{wsc}}$. Let $p_1, \ldots, p_{l'}$ be polynomials with non-negative coefficients such that $|g_j(\vec{b})| \leqslant p_j(|\vec{b}|)$ for each $j = 1, \ldots, l'$. It is well known that there exists such a length-bounding polynomial for any polynomial time function. Define another polynomial q by $q(\vec{x}) = \sum_{j=1}^{l'} p_j(p(\vec{x}), \ldots, p(\vec{x}))$. Clearly the polynomial q has only non-negative coefficients. By IH for h there exists a witnessing function $\{h, q\}$. We define a function $\{f, p\} \in \mathcal{B}_{\mathsf{wsc}}$ by $\{f, p\}(\vec{x}; \vec{b}) = \{h, p\}(\vec{x}; \{g_1, p\}(\vec{x}; \vec{b}), \ldots, \{g_{l'}, p\}(\vec{x}; \vec{b}))$. Suppose that $\max|\vec{b}| \leqslant p(|\vec{x}|)$ holds. Then for each $j = 1, \ldots, l'$ we have

$$\begin{aligned}
|\{g_j, p\}(\vec{x}; \vec{b})| &= |g_j(\vec{b})| && \text{(by IH for } g_j) \\
&\leqslant p_j(|\vec{b}|) \\
&\leqslant p_j(p(|\vec{x}|), \ldots, p(|\vec{x}|)) && \text{(by monotonicity of } p_j) \\
&\leqslant q(|\vec{x}|) .
\end{aligned}$$

Hence the following equality holds:

$$\begin{aligned}
\{f, p\}(\vec{x}; \vec{b}) &= \{h, p\}(\vec{x}; \{g_1, p\}(\vec{x}; \vec{b}), \ldots, \{g_{l'}, p\}(\vec{x}; \vec{b})), \\
&= h(\{g_1, p\}(\vec{x}; \vec{b}), \ldots, \{g_{l'}, p\}(\vec{x}; \vec{b})) && \text{(by IH for } h) \\
&= h(g_1(\vec{b}), \ldots, g_{l'}(\vec{b})) && \text{(by IH for } g_1, \ldots, g_{l'}) \\
&= f(\vec{b}) .
\end{aligned}$$

Case. f is defined by polynomially length-bounded recursion on notation from some polynomial time functions g, h_0, h_1 and some length-bounding polynomial p_f by

$$\begin{aligned}
f(\epsilon, \vec{b}) &= g(\vec{b}), \\
f(ai, \vec{b}) &= h_i(a, \vec{b}, f(a, \vec{b})) && (i = 0, 1) \\
|f(a, \vec{b})| &\leqslant p_f(|a|, |\vec{b}|) :
\end{aligned}$$

Let an arbitrary polynomial p with non-negative coefficients be given. By IH for g there exists a witnessing function $\{g, p\} \in \mathcal{B}_{\mathsf{wsc}}$. Define a polynomial q by $q(\vec{x}) = p(\vec{x}) + p_f(p(\vec{x}), \ldots, p(\vec{x}))$. By IH for h_0, h_1 there exist witnessing functions $\{h_i, q\} \in \mathcal{B}_{\mathsf{wsc}}$ for each $i = 0, 1$. In order to define a witnessing function $\{f, p\} \in \mathcal{B}_{\mathsf{wsc}}$ we define auxiliary functions $\langle h_0, q\rangle$ and $\langle h_1, q\rangle$ by $\langle h_i, q\rangle(\vec{x}, a, \vec{b}, c) = \{h_i, q\}(\vec{x}; a, \vec{b}, c)$ $(i = 0, 1)$. Further we define another auxiliary function $\langle f, p\rangle$ by

$$\begin{aligned}
\langle f, p\rangle(\vec{x}, \varepsilon, \vec{b}, c, d) &= c, \\
\langle f, p\rangle(\vec{x}, ai, \vec{b}, c, d) &= \langle h_i, p\rangle(\vec{x}, d \oplus a, \vec{b}, \langle f, p\rangle(\vec{x}, a, \vec{b}, c, d)). && (i = 0, 1)
\end{aligned}$$

By definition $\langle f, p\rangle$ and $\langle h, q\rangle$ meet the conditions in the premise of Lemma 15. Hence by Lemma 16 we have a function $\langle f, p\rangle[p] \in \mathcal{B}_{\mathsf{wsc}}$ which witnesses Lemma 16 on $\langle f, p\rangle$. Now we define the function $\{f, p\} \in \mathcal{B}_{\mathsf{wsc}}$ by $\{f, p\}(\vec{x}; a, \vec{b}) = \langle f, p\rangle[p](\vec{x}; a, \vec{b}, \{g, p\}(\vec{x}; \vec{b}), \epsilon)$. Suppose that $\max(|a|, |\vec{b}|) \leqslant p(|\vec{x}|)$ holds. We show that $\{f, p\}(\vec{x}; a, \vec{b}) = f(\vec{b})$ holds by (side) induction on $|a|$. In the base case the following equality holds:

$$\begin{aligned}
\{f,p\}(\vec{x};\epsilon,\vec{b}) &= \langle f,p\rangle[p](\vec{x};\epsilon,\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon) \\
&= \langle f,p\rangle(\vec{x},\min(\epsilon,\oplus_p(\vec{x};\epsilon)),\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon) && \text{(by Lemma 16)} \\
&= \langle f,p\rangle(\vec{x},\epsilon,\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon) \\
&= \{g,p\}(\vec{x};\vec{b}) \\
&= g(\vec{b}) && \text{(by IH for } g\text{)} \\
&= f(0,\vec{b})\ .
\end{aligned}$$

In the induction case $\max(|a|,|\vec{b}|) \leqslant p(|\vec{x}|)$ holds since we are assuming that $\max(|a|+1,|\vec{b}|) \leqslant p(|\vec{x}|)$ holds. Hence the following equality holds:

$$\begin{aligned}
&\{f,p\}(\vec{x};ai,b) \\
&\quad= \langle f,p\rangle[p](\vec{x};a_i,\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon), \\
&\quad= \langle f,p\rangle(\vec{x},\min(|a_i|,\oplus_p(\vec{x};\epsilon)),\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon) && \text{(by Lemma 16)} \\
&\quad= \langle f,p\rangle(\vec{x},a_i,\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon) && \text{(as } \max(|a|+1,|\vec{b}|) \leqslant p(|\vec{x}|)\text{)} \\
&\quad= \langle h_i,q\rangle(\vec{x},a,\vec{b},\langle f,p\rangle(\vec{x},a,\vec{b},\{g,p\}(\vec{x};\vec{y}),\epsilon)) \\
&\quad= \langle h_i,q\rangle(\vec{x},a,\vec{b},\langle f,p\rangle(\vec{x},\min(a,\oplus_p(\vec{x};\epsilon)),\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon)) \\
&\quad= \langle h_i,q\rangle(\vec{x},a,\vec{b},\langle f,p\rangle[p](\vec{x};a,\vec{b},\{g,p\}(\vec{x};\vec{b}),\epsilon)) && \text{(by Lemma 16)} \\
&\quad= \langle h_i,q\rangle(\vec{x},a,\vec{b},\{f,p\}(\vec{x};a,\vec{b})) = \langle h_i,q\rangle(\vec{x},a,\vec{b},f(a,\vec{b})) && \text{(by SIH)} \\
&\quad= \{h_i,q\}(\vec{x};a,\vec{b},f(a,\vec{b})) = h_i(a,\vec{b},f(a,\vec{b})) && \text{(by IH for } h_i\text{)} \\
&\quad= f(ai,\vec{b})\ .
\end{aligned}$$

This completes the case and hence the proof of the lemma. ☐

Proof. Let f be a k-ary polynomial-time computable function. Define a k-ary polynomial p with non-negative coefficients by $p(\vec{x}) = x_1 + \cdots + x_k$. Then there exists a function $\{f,p\} \in \mathcal{B}_{\mathsf{wsc}}$ which witnesses Lemma 18. By the definition of the polynomial p, $\max|\vec{x}| \leqslant p(|\vec{x}|)$ holds. Hence by Lemma 18 the equality $\{f,p\}(\vec{x};\vec{x}) = f(\vec{x})$ holds. This implies that $f \in \bigcup_{k\in\mathbb{N}} \mathcal{B}^{k,0}_{\mathsf{wsc}}$. ☐

9 A Non-Trivial Closure Property of the Polytime Functions

In this section we introduce *small polynomial path order with parameter substitution* ($\mathrm{sPOP}^*_{\mathrm{PS}}$ for short), that extends clause $>^{\langle 2\rangle}_{\mathsf{spop}*}$ to account for parameter substitution.

Definition 19. *Let s and t be terms such that $s = f(s_1,\ldots,s_k\,;s_{k+1},\ldots,s_{k+l})$. Then $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$ if one of the following alternatives holds.*

1. *$s_i \geqslant_{\mathsf{spop}^*_{\mathsf{ps}}} t$ for some argument s_i of s.*
2. *f is a defined symbol, $t = g(t_1,\ldots,t_m\,;t_{m+1},\ldots,t_{m+n})$ such that g is below f in the precedence and the following conditions hold:*

   a) $s \rhd_{\mathsf{n}} t_j$ *for all normal arguments* t_j *of* t*;*
   b) $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ *for all for all normal arguments* t_j *of* t*;*
   c) *except for one argument* t_{j_0}*, all arguments* t_j $(j \neq j_0)$ *contain only function symbols below* f *in the precedence.*

3. f *is recursive and* $t = g(t_1, \dots, t_k ; t_{k+1}, \dots, t_{k+m})$ *such that* g *is equivalent to* f *in the precedence and the following conditions hold:*
   a) $\langle s_1, \dots, s_k\rangle >_{\mathsf{spop}^*_{\mathsf{ps}}} \langle t_{\pi(k)}, \dots, t_{\pi(k)}\rangle$ *for some permutation* π *on normal argument positions;*
   b) $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t_j$ *for all safe arguments* t_j*;*
   c) *all safe arguments* t_j *contain only function symbols below* f *in the precedence.*

Here $s \geqslant_{\mathsf{spop}^*_{\mathsf{ps}}} t$ *denotes that either* s *and* t *are equivalent or* $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$*. In the last clause, we use* $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ *also for the product extension of* $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ *(modulo permutation).*

As evident from our experiments, parameter substitution extends the analytical power of sPOP* significantly. In particular, sPOP* can handle tail recursion as in the TRS $\mathcal{R}_{\mathsf{rev}}$ whose rules are depicted in Fig. 7. Whereas $\mathcal{R}_{\mathsf{rev}}$ cannot be handled by sPOP*, it is compatible with $>_{\mathsf{spop}^*_{\mathsf{ps}}}$ as induced by the precedence $\mathsf{rev} > \mathsf{rev}_{\mathsf{tl}} > :$ where only $\mathsf{rev}_{\mathsf{tl}}$ is recursive.

Still sPOP$^*_{\mathrm{PS}}$ induces polynomially bounded runtime complexity in the sense of Theorem 2. We emphasise that the proof requires only minor modification. First, we verify that the set $\mathcal{T}_{\mathsf{b}}^{\rightarrow}$ is closed under rewriting in the sense of Lemma 4.

Lemma 20. *Let* $\mathcal{R}$ *be a completely defined TRS compatible with* $>_{\mathsf{spop}^*_{\mathsf{ps}}}$*. If* $s \in \mathcal{T}_{\mathsf{b}}^{\rightarrow}$ *and* $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ *then* $t \in \mathcal{T}_{\mathsf{b}}^{\rightarrow}$*.*

Proof. The Lemma follows by a straight forward inductive argument on Definition 19. □

Further, innermost rewrite step embed into $\blacktriangleright_k$ in accordance to Lemma 11.

Lemma 21. *Let* $\mathcal{R}$ *be a completely defined TRS compatible with* $>_{\mathsf{spop}^*_{\mathsf{ps}}}$*. Let* $\ell \geqslant \max\{|r| \mid l \to r \in \mathcal{R}\}$*. If* $s \in \mathcal{T}_{\mathsf{b}}^{\rightarrow}$ *and* $s \xrightarrow{\mathsf{i}}_{\mathcal{R}} t$ *then* $\mathsf{S}(s) \blacktriangleright_{\ell} \mathsf{S}(t)$*.*

Proof. The proof follows the proof steps of Lemma 11. The only deviation is that the application of the auxiliary Lemma 10 is replaced by the stronger statement: If $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$ then (i) $f_{\mathsf{n}}(s_1\sigma, \dots, s_l\sigma) \blacktriangleright_{|t|} u$ for all $u \in \mathsf{S}(t\sigma)$ and further, (ii) at most one $u \in \mathsf{S}(t\sigma)$ contains symbols not below f in the precedence. Here $s = f(s_1, \dots, s_l ; s_{l+1}, \dots, s_{l+m})$ is a basic term, and σ is a substitution that maps variables to values. Property (ii) follows again by straight

$$\mathsf{rev}(xs) \to \mathsf{rev}_{\mathsf{tl}}(xs, \mathsf{nil}) \qquad \mathsf{rev}_{\mathsf{tl}}(\mathsf{nil}, ys) \to ys$$
$$\mathsf{rev}_{\mathsf{tl}}(x : xs, ys) \to \mathsf{rev}_{\mathsf{tl}}(xs, x : ys)$$

Figure 7: Rewrite system $\mathcal{R}_{\mathsf{rev}}$.

Parameter Substitution ($\mathsf{SRN_{PS}}$)

$$f(0,\vec{x};\vec{y}) = g(\vec{x};\vec{y})$$
$$f(S_i(;z),\vec{x};\vec{y}) = h_i(z,\vec{x};\vec{y},f(z,\vec{x};\vec{p}(\vec{z},\vec{x};\vec{y})))\ (i=0,1)$$

Figure 8: Safe recursion on notation with parameter substitution

forward inductive reasoning, for Property (ii) the only new case is when $s >_{\mathsf{spop}^*_{\mathsf{ps}}} t$ follows by $>^{\langle 2\rangle}_{\mathsf{spop}^*_{\mathsf{ps}}}$. Consider $t = g(t_1,\dots,t_k\,;t_{k+1},\dots,t_{k+n})$ where

$$\mathsf{S}(t\sigma) = g_{\mathsf{n}}(t_1\sigma,\dots,t_k\sigma)\frown\mathsf{S}(t_{k+1}\sigma)\frown\cdots\frown\mathsf{S}(t_{k+n}\sigma)\ .$$

Exactly as in Lemma 10 we verify (‡). Unlike for the case $>^{\langle 2\rangle}_{\mathsf{spop}*}$, we cannot reason that the safe arguments t_j of t ($j = k+1,\dots,k+n$) are values. Instead, we use the induction hypothesis on t_j to conclude $f_{\mathsf{n}}(l_1\sigma,\dots,l_m\sigma) \blacktriangleright_\ell u$ for all $u \in \mathsf{S}(t_j\sigma)$. □

Theorem 22. *Let $\mathcal{R}$ denote a predicative recursive TRS of degree d. Then the innermost derivation height of any basic term $f(\vec{u}\,;\vec{v})$ is bounded by a polynomial of degree d in the sum of the depths of normal arguments $\vec{u}$.*

Proof. The proof goes in accordance to the proof of Theorem 2, where we replace the application of Lemma 11 and Lemma 4 by the corresponding Lemma 21 and Lemma 20 respectively. □

The order $\mathrm{sPOP}^*_{\mathrm{PS}}$ is complete for the class of polytime computable functions. However, in order to state a stronger completeness result, we introduce an extension of the class $\mathcal{B}_{\mathsf{wsc}}$ according to the definition of $\mathrm{sPOP}^*_{\mathrm{PS}}$. Let $\mathcal{B}_{\mathsf{wsc+ps}}$ denote the smallest class containing $\mathcal{B}_{\mathsf{wsc}}$ and closed under weak safe composition (WSC) and *safe recursion on notation with parameter substitution* ($\mathsf{SRN_{PS}}$) which is presented in Fig. 8. Then $\mathrm{sPOP}^*_{\mathrm{PS}}$ is complete for $\mathcal{B}_{\mathsf{wsc+ps}}$ in the same sense as Theorem 13. Here we adapt the notion of predicative recursive TRS of degree d to $\mathrm{sPOP}^*_{\mathrm{PS}}$ in the obvious way.

Theorem 23. *For any $\mathcal{B}_{\mathsf{wsc+ps}}$-function f there exists a confluent TRS $\mathcal{R}_f$ that that is predicative recursive of degree d, where d equals the maximal number of nested application of ($\mathsf{SRN_{PS}}$) in the definition of f.*

Proof. The proof goes in accordance to the proof of Theorem 13. One will extend the TRS $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc}}}$ for $\mathcal{B}_{\mathsf{wsc}}$ to a TRS $\mathcal{R}_{\mathcal{B}_{\mathsf{wsc+ps}}}$ for $\mathcal{B}_{\mathsf{wsc+ps}}$ by adding rules corresponding to the schema of ($\mathsf{SRN_{PS}}$). Clearly this schema is a syntactic extension of the schema of (SRN). Hence we can replace the case of (SRN) by ($\mathsf{SRN_{PS}}$) and application of Definition 1.3 by application of Definition 19.3. □

Corollary 24. *The class $\mathcal{B}_{\mathsf{wsc}}$ is closed under predicative recursion with parameter substitution.*

Proof. By the theorem the extension of $\mathcal{B}_{\mathsf{wsc}}$ with the schema ($\mathsf{SRN_{PS}}$) yields only functions that are representable by predicative recursive TRS of degree. Thus these functions are polytime computable and due to Lemma 12 contained in $\mathcal{B}_{\mathsf{wsc}}$. □

bound	MPO	LMPO	POP*	sPOP*	POP$^*_{\mathrm{PS}}$	sPOP$^*_{\mathrm{PS}}$
$\mathsf{O}(1)$				9\0.06		9\0.06
$\mathsf{O}(n^1)$				32\0.07		46\0.09
$\mathsf{O}(n^2)$				38\0.09		53\0.10
$\mathsf{O}(n^3)$				39\0.20		54\0.22
$\mathsf{O}(n^k)$			43\0.05	39\0.20	56\0.05	54\0.22
yes	76\0.09	57\0.05	43\0.05	39\0.07	56\0.05	54\0.08
maybe	681\0.16	700\0.11	714\0.11	718\0.11	701\0.11	703\0.11

Table 1: Experimental Results

10 Experimental Results

The complexity analyser TCT features a fully automatic implementation of sPOP* and sPOP$^*_{\mathrm{PS}}$. To facilitate an efficient synthesis of a concrete order, we make use of the state-of-the-art SAT-solver MiniSAT [10]. The experiments were conducted on a laptop with 4Gb of RAM and Intel® Core™ i7-2620M CPU (2.7GHz).

In Table 1 we contrast the different orders on our testbed. The testbed is a subset of 757 examples from the termination problem database, version 8.0[3]. This subset was obtained by restricting the runtime complexity problem set to constructor TRSs, additionally removing TRSs that are not wellformed.[4].

The rows of Table 1 reflect the assessed bounds on the innermost runtime complexity. Additionally we annotate for each method the number of systems that were proven in total (row yes), and the number of systems were a proof was not obtained (row maybe). Since recursive path orders (with multiset status) (*MPO* for short) encompass LMPO as well as (small) polynomial path orders, we also included MPO for comparison. This reveals that predicative recursion limits the power of our techniques by roughly one fourth on our testbed. Comparing POP* with sPOP* we see an increase in precision accompanied with only minor decrease in power. Of the four systems that can be handled by POP* but not by sPOP*, two fail to be oriented because sPOP* weakens the multiset status to product status, and two fail to be oriented because of the weakening of the composition scheme. Compared to LMPO, polynomial path orders loose in power as they cannot deal with multiple recursive calls. Note that not all systems proven by LMPO admit polynomial (innermost) runtime complexity. The last two columns of Table 1 demonstrate that parameter substitution almost closes the gap in power to LMPO. Whether this extension is also possible for LMPO remains currently unknown.

[3] Available at `http://termcomp.uibk.ac.at/status/downloads/tpdb-8.0.tar.gz`

[4] Cf. `http://cl-informatik.uibk.ac.at/software/tct/experiments/lics2011` for full experimental evidence.

11 Conclusion

We propose a new order, the small polynomial path order sPOP*. Based on sPOP*, we delineate a class of rewrite systems, dubbed systems of predicative recursion of degree d, such that for rewrite systems in this class we obtain that the runtime complexity lies in $O(n^d)$. This is a tight characterisation in the sense that we can provide a family of systems of predicative recursion of depth d, such that their runtime complexity is bounded from below by $\Omega(n^d)$.

In future work we want to integrate sPOP* with the weak dependency pair framework that lifts the dependency pair method to the runtime complexity analysis. Furthermore, we aim to clarify the question whether the class of predicative recursive TRS of degree d *exactly* characterise those functions definable with d nested applications of safe recursion. We conjecture that the answer is yes, but further research in this direction is required. In [19] a type system for a simple imperative programming language is proposed that induces polytime computability. The definition of this type system is closely connected to predicative recursion. We want to investigate whether a similar type system can be crafted on the basis of the class $\mathcal{B}_{\mathsf{wsc}}$ studied in this paper. Perhaps such a study allows to certify more precise time bounds in the spirit of the class of predicative recursive TRSs of degree d.

References

[1] E. Albert, P. Arenas, S. Genaim, M. Gómez-Zamalloa, G. Puebla, D. Ramírez, G. Román, and D. Zanardini. Termination and cost analysis with costa and its user interfaces. *ENTCS*, 258(1):109–121, 2009.

[2] M. Avanzini and G. Moser. Complexity Analysis by Rewriting. In *Proc. of 9th FLOPS*, volume 4989 of *LNCS*, pages 130–146, 2008.

[3] M. Avanzini and G. Moser. Closing the Gap Between Runtime Complexity and Polytime Computability. In *Proc. of 21st RTA*, volume 6 of *LIPIcs*, pages 33–48, 2010.

[4] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[5] P. Baillot, J.-Y. Marion, and S. Ronchi Della Rocca. Guest editorial: Special Issue on Implicit Computational Complexity. *TOCL*, 10(4), 2009.

[6] A. Beckmann and A. Weiermann. A term rewriting characterization of the polytime functions and related complexity classes. *Arch. Math. Log.*, 36:11–30, 1996.

[7] S. Bellantoni and S. Cook. A new Recursion-Theoretic Characterization of the Polytime Functions. *CC*, 2(2):97–110, 1992.

[8] G. Bonfante, A. Cichon, J.-Y. Marion, and H. Touzet. Algorithms with Polynomial Interpretation Termination Proof. *JFP*, 11(1):33–53, 2001.

[9] A. Cobham. The Intrinsic Computational Difficulty of Functions. In *Proc. of 1964 LMPS*, pages 24–30, 1964.

[10] N. Eén and N. Sörensson. An Extensible SAT-solver. In *Proc. of 6th SAT*, volume 2919 of *LNCS*, pages 502–518, 2003.

[11] S. Gulwani, K.K. Mehra, and T.M. Chilimbi. Speed: precise and efficient static estimation of program computational complexity. In *Proc. of 36th POPL*, pages 127–139. ACM, 2009.

[12] W. G. Handley and S. S. Wainer. Complexity of Primitive Recursion. In U. Berger and H. Schwichtenberg, editors, *Computational Logic, NATO ASI Series F: Computer and Systems Science*, volume 165, pages 273–300. Springer, 1999.

[13] N. Hirokawa and G. Moser. Automated Complexity Analysis Based on the Dependency Pair Method. In *Proc. of 4th IJCAR*, volume 5195 of *LNAI*, pages 364–380, 2008.

[14] N. Hirokawa and G. Moser. Automated complexity analysis based on the dependency pair method. *CoRR*, abs/1102.3129, 2011. submitted.

[15] D. Hofbauer. Termination Proofs by Multiset Path Orderings Imply Primitive Recursive Derivation Lengths. *TCS*, 105(1):129–140, 1992.

[16] J. Hoffmann, K. Aehlig, and M. Hofmann. Multivariate amortized resource analysis. In *Proc. of 38th POPL*, pages 357–370. ACM, 2011.

[17] D. Leivant. A foundational delineation of computational feasiblity. In *Proc. of 6ht LICS*, pages 2–11. IEEE Computer Society, 1991.

[18] J.-Y. Marion. Analysing the Implicit Complexity of Programs. *IC*, 183:2–18, 2003.

[19] J.-Y. Marion. A type system for complexity flow analysis. In *Proc. of 26th LICS*, pages 123–132. IEEE Computer Society, 2011.

[20] Jean-Yves Marion. On tiered small jump operators. *Logical Methods in Computer Science*, 5(1), 2009.

[21] A. Middeldorp, G. Moser, F. Neurauter, J. Waldmann, and H. Zankl. Joint spectral radius theory for automated complexity analysis of rewrite systems. In *Proc. of 4th CAI*, volume 6472 of *LNCS*, pages 1–20, 2011.

[22] G. Moser. Proof Theory at Work: Complexity Analysis of Term Rewrite Systems. *CoRR*, abs/0907.5527, 2009. Habilitation Thesis.

[23] L. Noschinski, F. Emmes, and J. Giesl. A dependency pair framework for innermost complexity analysis of term rewrite systems. In *Proc. of 23rd CADE*, LNCS, pages 422–438, 2011.

[24] H. Simmons. The realm of primitive recursion. *Arch. Math. Log.*, 27:177–188, 1988.